UNIVERSITY OF ALBERTA

# PATH RESTORABLE NETWORKS

By ©

RAINER R. IRASCHKO

A thesis submitted to the Faculty of Graduate Studies and
Research in partial fulfilment of the requirements for the degree
of Doctor of Philosophy

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

EDMONTON, ALBERTA

Spring 1997

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-34872-5

Canada

**UNIVERSITY OF ALBERTA**

**Library Release Form**

| | |
|---|---|
| **Name of Author:** | RAINER R. IRASCHKO |
| **Title of Thesis:** | PATH RESTORABLE NETWORKS |
| **Degree:** | DOCTOR OF PHILOSOPHY |
| **Year this Degree Granted:** | 1997 |

Permission is hereby granted to the University of Alberta to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly, or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion maybe printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Rainer R. Iraschko
Rainer R. Iraschko

69 Johnson Street

Thonhill, Ontario

L3T 2N9

Date: Dec 5, 1996

To see a World in a Grain of Sand
And a Heaven in a Wild Flower
Hold Infinity in the palm of your hand
And Eternity in an hour[1]

---

[1] William Blake, *Auguries of Innocence*

## UNIVERSITY OF ALBERTA

### Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled

**"Path Restorable Networks"**

submitted by **Rainer R. Iraschko** in partial fulfilment of the requirements for the degree of **Doctor of Philosophy.**

Wayne D. Grover (Supervisor)

Mike MacGregor

Mike H. MacGregor (Co-Supervisor)

Dave Allen (University External Examiner)

Joe Culberson

Jan Conradi

Jan Conradi

Bruce Cockburn

J. Harms

Janelle Harms

Date: Nov. 15, 1996

# Abstract

Intense competition between transport network service providers and the widespread deployment of vulnerable high-capacity fiber optic transport facilities has created the need for capacity-efficient transport networks with short restoration times. Ensuring service continuity affordably and quickly is called the restoration problem in modern telecommunications practice. Solving the restoration problem not only requires restoring a failure quickly, but determining an economic capacity placement which permits full restoration. This thesis solves the restoration problem in path restorable mesh networks by formulating a capacity design methodology, and an optimized distributed real time path restoration mechanism, named OPRA.

One of the main contributions of this thesis is OPRA. OPRA synthesizes link-disjoint loop-free pathsets that are very close to the multicommodity, max-flow ideal by autonomous, database-free, self-organizing interaction between nodes. The results presented in this thesis show that OPRA will in practice very likely restore a failure in a tightly spared network in less than two seconds, regardless of a network's size and topology, the distribution of alarms in time, and the number and location of connected demand pairs affected by a failure.

The second main contribution of this thesis is the capacity design methodology which uses integer programming. The integer program formulated in this thesis uses flow constraints based on a suitable set of predefined routes over which pathsets are implemented to optimize the spare and/or working capacity of a path restorable network. The results presented in this thesis show mesh restorable networks using path restoration are the most capacity efficient, reducing the total capacity of the corresponding span restorable network design by up to 19%.

Together these advances make feasible path restorable transport networks which restore failures within two seconds and require less spare capacity than existing transport networks. In the envisaged network, capacity would be minimized using the integer program, and failures would be restored in real time in an entirely autonomous distributed manner using OPRA.

# Acknowledgements

During the past four years many people have encouraged and helped me through my studies. There are so many individuals I'd like to acknowledge, it would be impossible for me to list everyone on this page. Those of my friends not mentioned here should know that their support was appreciated.

To begin, I want to thank my family. My family is my foundation which keeps me strong and allows me to face life cheerfully. My Mom and Dad taught me that there is nothing I can't do. Their love and support gives me the strength and confidence I need to be successful. I also want to thank my sister and "brother", Elke and Ed, who helped me keep life in perspective with weekend escapes "*to* reality" in the form of skiing and scrambling.

When I came to TRLabs four years ago my intent was to complete a M.Sc. The prospect of doing a Ph.D. was intimidating and foreboding. Had Dr. Grover not recognized my ability to complete a doctoral degree I would not be here today. I want to thank Dr. Grover for his guidance during the past four years as my supervisor. I also want to thank my co-supervisor, and paddling buddy, Dr. MacGregor. I also drew upon his knowledge and advice frequently, which sometimes encouraged me to go windsurfing so that I would maintain a healthy balance between work and play.

Besides my supervisors, I want to thank all of the people at TRLabs. TRLabs is a great place to study because of the people that work there. Jim Slevinsky whose musical anecdotes made me laugh when I was stressed, deserves special mention. I also want to acknowledge Nona McDonagh and Corinna Johnson who put up with me. I want to thank the TRLabs gang: Jason "Stubby" Palm, Master Demetrious "Lou" Stamatelakis, Master Danny Yau Chung Li, Dr. Ping Wan, and Master Ashish Duggal, who made working at TRLabs enjoyable. I also want to mention Kent Ryhorchuk, who helped me obtain the IP results presented in chapter 4, and became a very good friend.

In addition to my friends at TRLabs, I had the support of Dana Budnyk and David Carr from the University of Alberta Ski Team. I also want to thank my good friends Allan and Claudia Chow, Scott and Margret Mandel, Dean Michaels, and Jorge Nogera for their support.

Finally I want to mention my girlfriend Sue Lucenko. It is appropriate I thank her last because she walked into my life in the final year of my studies during the most difficult time. Her love, patience, and care helped me endure to the end.

# Table of Contents

**PART I: Introduction and Capacity Placement Methodology** .............................. 1

Chapter 1. Introduction and Background ........................................................ 2
- 1.1 The Restoration Problem ........................................................ 2
- 1.2 Economic Impact of Outages ........................................................ 2
  - 1.2.1 Restoration Time Objectives ........................................................ 3
  - 1.2.2 Capacity Efficiency Objectives ........................................................ 5
- 1.3 Present Methods for Transport Network Restoration .............................. 5
  - 1.3.1 Automatic Protection Switching ........................................................ 6
  - 1.3.2 Self-Healing Rings ........................................................ 8
- 1.4 Mesh Restoration of Transport Networks ........................................................ 11
  - 1.4.1 Centralized vs. Distributed Restoration ........................................ 12
  - 1.4.2 Span vs. Path Restoration ........................................................ 13
  - 1.4.3 Preplanned vs. Dynamic Restoration ........................................ 15
  - 1.4.4 Optimized Distributed Dynamic Path Restoration ........................ 17
- 1.5 Capacity Placement for Mesh Survivable Networks ................................ 20
  - 1.5.1 Using Heuristics to Solve the Capacity Placement Problem ...... 20
  - 1.5.2 Using Integer Programs to Solve the Capacity Placement Problem ........................................................ 22
- 1.6 Research Contributions of this Thesis ........................................................ 26
- 1.7 Outline of Thesis ........................................................ 27

Chapter 2. Transport Networks ........................................................ 29
- 2.1 SONET ........................................................ 29
  - 2.1.1 Distributing Alarms in a SONET Network ........................................ 32
  - 2.1.2 SONET Data Communications Channel (DCC) ........................ 35
  - 2.1.3 State-based Signalling in a SONET Network ............................ 36
  - 2.1.4 SONET DCSs ........................................................ 37
- 2.2 Relationship to ATM ........................................................ 41

Chapter 3. Restoration as a Routing Problem ....................45
3.1 Describing Transport Networks using Graph Theoretical Terms ....................45
3.2 Relating Path Restoration to Call and Data Routing ....................47
3.3 Routing Formulation for Path Restoration ....................49
3.4 Complexity Considerations ....................53

Chapter 4. Capacity Placement in Mesh Restorable Networks ....................56
4.1 Lower Bounds on Spare Capacity Requirements in Span and Path Restorable Networks. ....................56
4.2 Integer Program Formulation ....................62
4.2.1 Integer Program Formulation for Spare Capacity Placement in Networks with Pre-defined Demand Routing ....................62
4.2.2 Integer Program Formulation for Combined Optimized Spare and Working Capacity Placement ....................66
4.3 Networks Investigated ....................68
4.4 Capacity Placement Test Results ....................71
4.5 Discussion of Capacity Placement Results ....................79
4.5.1 Capacity Savings of Path Restoration in Networks with Pre-defined Demand Routing ....................79
4.5.2 Capacity Savings due to Joint Working and Spare Optimization ....................81
4.5.3 Routing Effects of Joint Working and Spare Optimization ....................82
4.6 Summary of IP Network Designs ....................84
4.7 Conclusions ....................85

**PART II: Optimized Distributed Path Restoration Algorithm**....................86

Chapter 5 Performance Metrics and Theoretical Objectives for OPRA ....................87
5.1 Operational Performance Metrics ....................87
5.1.1 Span Restorability and Network Restorability ....................88
5.1.2 Speed Related Performance Measures ....................88

5.2 Intrinsic Path Metrics ....................90
    5.2.1 Path Number Efficiency (PNE) ....................91
    5.2.2 Path Length Efficiency (PLE) ....................92
5.3 Summary ....................93

Chapter 6. The Interference Heuristic for Coordinating Pathset Formation ....................94
6.1 The Interference Principle ....................94
    6.1.1 Validating the Interference Principle: A Centralized Path Restoration Algorithm ....................97
6.2 Distributed Implementation of a Path DRA based on the Interference Principle ....................103
    6.2.1 End-node Bottleneck Effect and the need for Bidirectional Flooding ....................110
6.3 Conclusion ....................112

Chapter 7. Distributed Interaction via Restoration Statelets ....................113
7.1 State Based Signalling via Restoration Statelets ....................113
7.2 Content and Uses of Restoration Statelet Information Fields ....................114
7.3 Factors Determining Statelet Length ....................123
7.4 Classifying Restoration Statelets ....................126
7.5 Relationships between Restoration Statelets ....................126
7.6 The Logical Environment of a Node ....................128
    7.6.1 Port Status Register ....................128
7.7 Summary ....................132

Chapter 8. Description of the Optimized Distributed Path Restoration Algorithm (OPRA) ....................134
8.1 Finite State Machine (FSM) Representation of OPRA ....................134
8.2 Initializing and Activating OPRA ....................137
    8.2.1 Initialization (State 11, *Initialize*) ....................137
    8.2.2 Idle State (State 14) ....................138

8.2.3 Receipt of an Alarm (State 7, *New_AIS*) ....................................139
8.3 Bidirectional Selective Forward Flooding ..........................................143
8.3.1 Forward Flood Management Logic (State 12, *Broadcast*) ..........144
8.3.2 Event Parsing at a Tandem Node (State 1, *Tandem_Node*) ......151
8.3.3 Arrival of a New Restoration Statelet at a Tandem Node (State 2, *New_Statelet*) ......................................153
8.3.4 Overwriting a Restoration Statelet at a Tandem Node (State 3, *Statelet_Ovwrt*) ....................................156
8.3.5 Disappearance of a Restoration Statelet at a Tandem Node (State 4, *Statelet_Vanish*) ..................................158
8.4 Recognizing a Match ..........................................................................158
8.5 Reverse Linking ..................................................................................163
8.5.1 Receiving a Reverse Linking statelet (State 5, *Complement_Statelet*) ................................................165
8.6 Restoration Path Confirmation ..........................................................169
8.6.1. The Arrival of a Statelet at a Destination Node (State 9, *Destination_Node*) .....................................................169
8.6.2 Processing a Forward Flooding or a Complemented Statelet at a Destination Node (State 8, *Receive_Statelet*) ......................170
8.6.3 Processing a Confirmation Statelet at a Tandem Node (State 6, *Confirmation*) ...........................................................174
8.6.4. Traffic Substitution (State 10, *Restore_Signal*) ..........................175
8.7 Terminating a Restoration Event .......................................................176
8.7.1 Cancelling Outstanding Statelets (State 13, *TimeOut*) ...............176
8.8 Network Level View ............................................................................177

Chapter 9. OPRA Implementation Details ................................................179
9.1 Preventing Looping Event Sequences .................................................179
9.2 Correct Sequencing of Port Updates ...................................................183
9.3 Re-Initiating Reverse Linking ..............................................................184
9.4 Optimizing the Forward Flooding Process ...........................................186

Chapter 10. Implementation of a Simulation Test-Bed for OPRA ........................................188
10.1 Modelling Environment ........................................188
10.1.1 Modelling OPRA ........................................189
10.1.2 The Testbed ........................................190
10.2 Testbed Description ........................................191
10.2.1 OPNET's Network Domain ........................................192
10.2.2 OPNET's Node Domain ........................................194
10.2.3 OPNET's Process Domain ........................................196
10.3 Port Card Process Model ........................................196
10.4 Polling Mechanism Process Model ........................................198
10.5 Initiating a Network Failure ........................................203
10.6 Collecting Working and Restoration Path Data ........................................203
10.7 Verifying OPRA's Operation ........................................205

Chapter 11. OPRA Test Results ........................................207
11.1 Test Networks ........................................207
11.2 Test Result Presentation Format ........................................208
11.3 Option Settings ........................................211
11.4 Experimental Results in a Metropolitan Network ........................................213
11.4.1 Optimized Spare Capacity Design for a Path Restorable Network without Stub Release ........................................213
11.4.2 Combined Capacity Design for a Path Restorable Network with Stub Release ........................................217
11.5 Experimental Results in a Long Haul Network ........................................221
11.5.1 Optimized Spare Capacity Design for a Path Restorable Network without Stub Release ........................................221
11.5.2 Combined Capacity Design for a Path Restorable Network with Stub Release ........................................225
11.6 Interpretation and Discussion of Results ........................................229
11.6.1 Restoration Trajectories ........................................230

11.6.2 Restoration Path Times ....................232
11.6.3 Restoration Path Lengths ....................233
11.6.4 Interference Numbers ....................234
11.6.5 Scatterplots ....................234
11.7 Summary of all OPRA Test Results ....................235
11.8 Conclusion ....................236

Chapter 12. Effects of Decreasing the Processing Delay ....................238
12.1 Study Network ....................238
12.2 Test Result Presentation Format ....................239
12.3 Option Settings ....................239
12.4 Experimental Results in Metropolitan Network without Stub Release ....................240
12.5 Interpretation of Results ....................243
12.5.1 Restoration Trajectories ....................243
12.5.2 Restoration Path Times ....................244
12.5.3 Restoration Path Lengths ....................244
12.5.4 Interference Numbers ....................244
12.5.5 Scatterplots ....................245
12.6 Conclusion ....................245

Chapter 13. Effects of Random Individual Link Failure Times ....................246
13.1 Staggered Alarms ....................246
13.2 Direct Exposure of OPRA to Staggered Alarms ....................247
13.3 Study Network ....................247
13.4 Test Result Presentation Format ....................247
13.5 Option Settings ....................248
13.6 Experimental Results in Metropolitan Network with Stub Release ....................248
13.7 Interpretation of Results ....................252
13.7.1 Working Path Failure Times ....................252
13.7.2 Restoration Trajectories ....................252

13.7.3 Restoration Path Times ....253

13.7.4 Restoration Path Lengths ....253

13.7.5 Interference Numbers ....254

13.7.6 Scatterplots ....254

13.8 Conclusion ....254

Chapter 14. Eliminating the Interference Heuristic In OPRA ....255

14.1 Eliminating the Interference Heuristic ....255

14.2 Study Network ....256

14.3 Test Result Presentation Format ....256

14.4 Option Settings ....257

14.5 Experimental Results in a Metropolitan Network without Stub Release ....257

14.6 Interpretation of Results ....260

14.6.1 Restoration Trajectories ....260

14.6.2 Restoration Path Times ....261

14.6.3 Restoration Path Lengths ....261

14.7 Conclusion ....262

Chapter 15. Achieving Restoration Levels Proportional to a Network's Pre-Failure Connectivity After Multiple Span Failures ....263

15.1 Network Recovery from Multiple Span Failures and Node Loss ....263

15.2 Study Network ....265

15.3 Test Result Presentation Format: Starplots ....265

15.4 Option Settings ....266

15.5 Experimental Results in Metropolitan Network with Stub Release ....267

15.5.1 IIN_STEP Set to Zero ....268

15.5.2 IIN_STEP Set to Fifty ....270

15.6 Interpretation and Discussion of Results ....272

15.7 Conclusion ....273

Chapter 16. Concluding Discussion ....................275
16.1 Review of Thesis ....................275
16.2 Summary of Research Results ....................280
16.3 Further Research ....................281
16.3.1 Implementing the Interference Heuristic in ATM Networks ....................281
16.3.2 Further Studies in Capacity Placement ....................283
16.3.3 Effects of Network Topology on Capacity Placement in Mesh Restorable Networks ....................283
16.3.4 Multi-Commodity Max-Flow Pathset Characterization ....................284
16.3.5 A New Approach to Avoiding the End-Node Bottleneck Traversal Problem ....................284
16.3.6 Alarm Distribution in the Event of Node or Multiple Simultaneous Span Failures ....................285
16.3.7 Reversion ....................286
16.3.8 Bypassing Tandem Nodes ....................286

# List of Tables

Chapter 2. Transport Networks

Table 2.1. Plesiochronous Digital Hierarchy ........................................30
Table 2.2. Levels of the SONET Signal Hierarchy........................................31
Table 2.3. Comparing Mesh Survivable SONET and ATM Networks ................43

Chapter 3. Restoration as a Routing Problem

Table 3.1. Comparative aspects of packet, call,
and restoration routing problems........................................49
Table 3.2. Computational complexity of various routing formulations................54

Chapter 4. Capacity Placement in Mesh Restorable Networks

Table 4.1. Test Network Characteristics ........................................68
Table 4.2. Capacity Design Case 1 ........................................75
Table 4.3. Capacity Design Case 2 ........................................76
Table 4.4. Capacity Design Case 3 ........................................76
Table 4.5. Capacity Design Case 4 ........................................77
Table 4.6. Capacity Design Case 5 ........................................77
Table 4.7. Capacity Design Case 6 ........................................78
Table 4.8. Standard deviation in span working capacity
confirming demand routing dispersion effect........................................82
Table 4.9. Average Working Path Lengths ........................................83
Table 4.10. Summary of IP Network Designs........................................84
Table 4.11. Summary of Spare Capacity Designs Normalized to Case 1..........84

Chapter 6. The Interference Heuristic for Coordinating End-to-End Pathset Formation

Table 6.1. Performance Studies of a Centralized Path Restoration Mechanism based on the Interference Heuristic ..............................101

Chapter 7. Distributed Interaction via Restoration Statelets

Table 7.1. Classifying Restoration Statelets .......................................................126

Chapter 11. OPRA Test Results

Table 11.1. Operational performance metrics....................................................229
Table 11.2. Intrinsic path metrics .......................................................................229
Table 11.3. Test Network Characteristics..........................................................230
Table 11.4. Summary of all OPRA Tests............................................................235

Chapter 15. Achieving Restoration Levels Proportional to a Network's Pre-Failure Connectivity After Multiple Span Failures

Table 15.1. Restored Capacity with IIN_STEP set to zero for metropolitan network with stub release after the failure of spans 9-10 & 5-7........269
Table 15.2. Restored Capacity with IIN_STEP set to zero for metropolitan network with stub release after the failure of spans 9-10 & 5-7......271

# List of Figures

**Chapter 1. Introduction and Background**

Figure 1.1. Restoration Time Impact ........ 4
Figure 1.2. 1:1 Automatic Protection Switching ........ 7
Figure 1.3. Types of Self-Healing Rings ........ 10
Figure 1.4. Span vs. Path Restoration ........ 14
Figure 1.5. Mesh Restoration Architectures ........ 16
Figure 1.6. Survivable Transport Network Architectures ........ 18
Figure 1.7. Example illustrating sub-optimality of "ad-hoc" path restoration ........ 19
Figure 1.8. Cutsets ........ 23
Figure 1.9. Predefined Routes ........ 25

**Chapter 2. Transport Networks**

Figure 2.1. Path Alarm Distribution in a SONET Network ........ 33
Figure 2.2. SONET DCS functional diagram with enhancements to support state-based signalling ........ 38
Figure 2.3. ATM VP-Based Network Architecture and Nodal Functions ........ 42

**Chapter 3. Restoration As A Routing Problem**

Figure 3.1. An anti-reflexive symmetric multigraph, i.e. a transport network ........ 46

**Chapter 4. Capacity Placement In Mesh Restorable Networks**

Figure 4.1. Average number of spans used per node to restore a failure ........ 59
Figure 4.2. Lower bounds on redundancy in mesh restorable networks ........ 60
Figure 4.3. Integer Program Notation ........ 63
Figure 4.4. Topology of Network 1 ........ 69
Figure 4.5. Topology of Network 2 ........ 69

Figure 4.6. Topology of Network 3 ........ 69
Figure 4.7. Topology of Network 4 ........ 70
Figure 4.8. Topology of Network 5 ........ 70
Figure 4.9. Network 1 Designs ........ 72
Figure 4.10. Network 2 Designs ........ 73
Figure 4.11. Network 3 Designs ........ 73
Figure 4.12. Network 4 Designs ........ 74
Figure 4.13. Network 5 Designs ........ 74
Figure 4.14. Optimization of Spare Capacity ........ 80

**Chapter 6. The Interference Heuristic for Coordinating End to End Pathset Formation**

Figure 6.1. Concept of Interference Numbers ........ 96
Figure 6.2. Program used to test the interference heuristic ........ 99
Figure 6.3 Span Interference Numbers ........ 106
Figure 6.4. Generating a cyclical path using negative span interference numbers ........ 108
Figure 6.5. Examples of basic target broadcast patterns ........ 109
Figure 6.6. Principle of End-Node Bottlenecking ........ 112

**Chapter 7. Distributed Interaction via Restoration Statelets**

Figure 7.1. Information fields in a statelet ........ 114
Figure 7.2. Indexing statelets ........ 117
Figure 7.3. Setting the Reverse Linking Indicator in a statelet ........ 120
Figure 7.4. Initiating a loop-back test using the Confirmation Indicator ........ 122
Figure 7.5. Precursor Relationships at Tandem Nodes ........ 127
Figure 7.6. Information fields in the PSReg ........ 129
Figure 7.7. Alarm detection related PSReg contents after the failure of a working path ........ 130

Chapter 8. Description of the Optimized Distributed Path Restoration Algorithm (OPRA)

Figure 8.1. OPRA State Transition Diagram ....................136
Figure 8.2. Identifying a severed precursor ....................140
Figure 8.3. Broadcasting precursors ....................146
Figure 8.4. Establishing a restoration path one hop long ....................148
Figure 8.5. Growing a statelet's broadcast mesh away from the source. ..........149
Figure 8.6. Computing the composite broadcast pattern at a node ....................150
Figure 8.7. Re-rooting a precursor ....................155
Figure 8.8. Rerooting a broadcast mesh after a precursor has changed positions ....................156
Figure 8.9. Problems when two tandem nodes complete a match simultaneously ....................160
Figure 8.10. Ignoring a match when a node is attempting to establish a restoration path one hop long ....................161
Figure 8.11. Avoiding "hair-pins" when forming restoration paths ....................162
Figure 8.12. Receiving a reverse linking statelet ....................165
Figure 8.13. Reverse linking ....................167
Figure 8.14. Initiating a loop-back test ....................171
Figure 8.15. Problems associated with anchoring a statelet's broadcast mesh at the source ....................172

Chapter 9. OPRA Implementation Details

Figure 9.1. Competition between statelets at a node ....................179
Figure 9.2. Using the handicap indicator to avoid oscillations ....................180
Figure 9.3. Correct sequencing of port updates ....................183
Figure 9.4. Overwriting a precursor from the same statelet family on a activated port ....................184
Figure 9.5. Initiating forward flooding statelets on different spares on a connecting span ....................187

**Chapter 10. Implementation of a Test-Bed for OPRA**

Figure 10.1. OPNET Network Domain specification for network number 1.......193
Figure 10.2. OPNET Node Domain specification for a DCS terminating four spans.......195
Figure 10.3. State transition diagram for process model path_DCS_ports.......197
Figure 10.4. Model of the polling mechanism used by OPRA.......198
Figure 10.5. State transition diagram for process model Timer.......200
Figure 10.6. Sample output generated by OAM node.......204

**Chapter 11. OPRA Test Results**

Figure 11.1. Restoration trajectories in a metropolitan network without stub release.......213
Figure 11.2. Distribution of restoration path times in a metropolitan network without stub release over all span cuts.......214
Figure 11.3. Distribution of restoration path lengths found by OPRA in a metropolitan network without stub release over all span cuts.......214
Figure 11.4. Distribution of restoration path lengths in the IP design for a metropolitan network without stub release over all span cuts.......215
Figure 11.5. Distribution of interference numbers in a metropolitan network without stub release over all span cuts.......215
Figure 11.6. Path time versus interference number for all span cuts in a metropolitan network without stub release.......216
Figure 11.7. Path length versus interference number for all span cuts in a metropolitan network without stub release.......216
Figure 11.8. Restoration trajectories in a metropolitan network with stub release.......217
Figure 11.9. Distribution of restoration path times in a metropolitan network with stub release over all span cuts.......218
Figure 11.10. Distribution of restoration path lengths found by OPRA in a metropolitan network with stub release over all span cuts.......218

Figure 11.11. Distribution of restoration path lengths in the IP design for a metropolitan network with stub release over all span cuts...........219
Figure 11.12. Distribution of interference numbers in a metropolitan network with stub release over all span cuts.................................219
Figure 11.13. Path time versus interference number for all span cuts in a metropolitan network with stub release ..........................220
Figure 11.14. Path length versus interference number for all span cuts in a metropolitan network with stub release ..........................220
Figure 11.15. Restoration trajectories in a long haul network without stub release.......................................................221
Figure 11.16. Distribution of restoration path times in a long haul network without stub release over all span cuts...........................222
Figure 11.17. Distribution of restoration path lengths found by OPRA in a long haul network without stub release over all span cuts ........222
Figure 11.18. Distribution of restoration path lengths in the IP design for a long haul network without stub release over all span cuts ...........223
Figure 11.19. Distribution of interference numbers in a long haul network without stub release over all span cuts...........................223
Figure 11.20. Path time versus interference number for all span cuts in a long haul network without stub release..........................224
Figure 11.21. Path length versus interference number for all span cuts in a long haul network without stub release..........................224
Figure 11.22. Restoration trajectories in a long haul network with stub release...........................................................225
Figure 11.23. Distribution of restoration path times in a long haul network with stub release over all span cuts................................226
Figure 11.24. Distribution of restoration path lengths found by OPRA in a long haul network with stub release over all span cuts .............226
Figure 11.25. Distribution of restoration path lengths in the IP design for a long haul network with stub release over all span cuts .............227

Figure 11.26. Distribution of interference numbers in a long haul network with stub release over all span cuts ........ 227
Figure 11.27. Path time versus interference number for all span cuts in a long haul network with stub release ........ 228
Figure 11.28. Path length versus interference number for all span cuts in a long haul network with stub release ........ 228

**Chapter 12. Effects of Decreasing the Processing Delay**

Figure 12.1. Restoration trajectories in a metropolitan network without stub release ........ 240
Figure 12.2. Distribution of restoration path times in a metropolitan network without stub release over all span cuts ........ 241
Figure 12.3. Distribution of restoration path lengths found by OPRA in a metropolitan network without stub release over all span cuts ...... 241
Figure 12.4. Distribution of interference numbers in a metropolitan network without stub release over all span cuts ........ 242
Figure 12.5. Path time versus interference number for all span cuts in a metropolitan network without stub release ........ 242
Figure 12.6. Path length versus interference number for all span cuts in a metropolitan network without stub release ........ 243

**Chapter 13. Effects of Random Individual Link Failure Times**

Figure 13.1. Distribution of alarms in a metropolitan network over all span cuts ........ 248
Figure 13.2. Restoration trajectories in a metropolitan network with stub release ........ 249
Figure 13.3. Distribution of restoration path times in a metropolitan network with stub release over all span cuts ........ 250
Figure 13.4. Distribution of restoration path lengths found by OPRA in a metropolitan network with stub release over all span cuts ........ 250

Figure 13.5. Distribution of interference numbers in a metropolitan network with stub release over all span cuts.................................251
Figure 13.6. Path time versus interference number for all span cuts in a metropolitan network with stub release .........................251
Figure 13.7. Path length versus interference number for all span cuts in a metropolitan network with stub release .........................252

Chapter 14. Eliminating the Interference Heuristic in OPRA

Figure 14.1. Restoration trajectories in a metropolitan network without stub release.........................................................258
Figure 14.2. Distribution of restoration path times in a metropolitan network without stub release over all span cuts...........................259
Figure 14.3. Distribution of restoration path lengths found by OPRA in a metropolitan network without stub release over all span cuts......259

Chapter 15. Achieving Restoration Levels Proportional to a Network's Pre-Failure Connectivity After Multiple Span Failures

Figure 15.1. Ideal star plot for metropolitan network with stub release after the failure of spans 9-10 and 5-7. .......................................266
Figure 15.2. Star plot with IIN_STEP set to zero for metropolitan network with stub release after the failure of spans 9-10 and 5-7. ...........268
Figure 15.3. Star plot with IIN_STEP set to fifty for metropolitan network with stub release after the failure of spans 9-10 and 5-7. ...........270

Chapter 16. Concluding Discussion

Figure 16.1. Suspending reverse linking as a result of bidirectional flooding ...............................................................285

# List of Abbreviations

| | |
|---|---|
| AAL | ATM Adaptation Layer |
| ADM | Add Drop Multiplexer |
| AIE | Alarm Interrupt Enable |
| AIS | Alarm Indication Signal |
| APS | Automatic Protection Switching |
| ATM | Asynchronous Transfer Mode |
| BER | Bit Error Rate |
| B-ISDN | Broadband Integrated Services Digital Network |
| BLSR | Bidirectional Line Self-healing Ring |
| CBR | Constant Bit Rate |
| CCSN | Common Channel Signalling Network |
| CDF | Cumulative Distribution Function |
| CPU | Central Processing Unit |
| CRC | Cyclical Redundancy Check |
| DCC | Data Communications Channel |
| DCS | Digital Crossconnect System |
| DMUX | Demultiplexer |
| DP | Diverse Protection |
| DRA | Distributed Restoration Algorithm |
| DS | Digital Signal |
| EMA | External Module Access |
| E/O | Electrical to Optical Converter |
| FCC | Federal Communications Committee |
| FERF | Far End Receive Failure |
| FSM | Finite State Machine |
| FOTS | Fibre Optic Transmission Systems |
| GOS | Grade Of Service |
| IIN | Initial Interference Number |

| | |
|---|---|
| IntNo | Interference Number |
| IP | Integer Program |
| ksp | k-successively-shortest link-disjoint paths |
| LAN | Local Area Network |
| LOF | Loss Of Frame |
| LOH | Line Overhead |
| LOS | Loss Of Signal |
| LP | Linear Program |
| LTE | Line Terminating Equipment |
| MUX | Multiplexer |
| NID | Node Identifier |
| NRR | Network Restoration Ratio |
| OAM | Operation and Management |
| OC | Optical Carrier |
| O/E | Optical to Electrical Converter |
| OLTM | Optical Line Terminating Multiplexer |
| OPRA | Optimized Distributed Path Restoration Algorithm |
| OSI | Open Systems Interconnection |
| PDH | Plesiochronous Digital Hierarchy |
| PDN | Plesiochronous Digital Network |
| PL | Private Line |
| PLE | Path Length Efficiency |
| PNE | Path Number Efficiency |
| POH | Path Overhead |
| PS | Protection Switch |
| PSReg | Port Status Register |
| PTE | Path Terminating Equipment |
| RA | Return Alarm |
| REG | Regenerator |
| RSCIE | Restoration Statelet Change Interrupt Enable |
| RxReg | Receive Register |

| | |
|---|---|
| RxRS | Receive Restoration Statelet |
| SDH | Synchronous Digital Hierarchy |
| SHN | Self-Healing Network (The span DRA described in [25]) |
| SHR | Self-Healing Rings |
| SOH | Section Overhead |
| SONET | Synchronous Optical Network |
| SPE | Synchronous Payload Envelope |
| STE | Section Terminating Equipment |
| STM | Synchronous Transfer Mode |
| STP | Signalling Transfer Point |
| STS | Synchronous Transport Signal |
| TDM | Time Division Multiplexing |
| TM | Terminal Multiplexer |
| TxReg | Transmit Register |
| VBR | Variable Bit Rate |
| VC | Virtual Circuit |
| VCI | Virtual Circuit Identifier |
| VP | Virtual Path |
| VPI | Virtual Path Identifier |
| X-25 | A packet-switched network access standard |

# List of Symbols

| | |
|---|---|
| $(u,v)$ | a span between nodes u and v |
| $(u_1, u_2, u_3, u_5)$ | a set of nodes defining a route |
| $[a_1, a_9, a_6]$ | a set of links defining a path |
| $[P]$ | path P as an ordered set of links |
| $\|[P]\|$ | the number of links in path P |
| $c(a_i)$ | the cost of link $a_i$ |
| $C([P])$ | the cost of path P, sometimes shortened to $C_P$ |
| $d$ | the average degree of a node |
| $d^r$ | the number of demand units between end-node pair r |
| $D_{xy}$ | the total number of demand pairs affected by span cut *xy*, sometimes shortened to $D_i$ where i is a tag for span *xy* |
| $D$ | the total number of non-zero demand pairs in the demand matrix |
| $f_i^{r,p}$ | he restoration flow through the p^th^ restoration route for demand pair r upon the failure of span i |
| $g^{r,q}$ | the working capacity required on the q^th^ working route to satisfy the demand between node pair r |
| $h(i)$ | is the logical path length of restoration path i. |
| $k_{xy}^{AZ}$ | the number of restoration paths used by relation A-Z to restore working links lost upon the failure of span x-y, sometimes shortened to $k_s^r$ where r is a tag for demand pair A-Z, and s is a tag for span *xy*. |
| $L_i^r$ | the restoration level required for demand pair r upon the failure of span i |

| | |
|---|---|
| n | the number of nodes in a network |
| O(f(x)) | the set of all functions bounded above by a positive real multiple of f(x), provided that x is sufficiently large. |
| P | path P as a vector of links |
| $P_i^r$ | the total number of eligible restoration routes for demand pair r upon the failure of span i |
| $Q^r$ | the total number of working routes available to satisfy the demand between node pair r |
| $r_l$ | a path defined as $[(A, u_{l,1}), s_{l,1}, l_{l,1,1}], [(u_{l,1}, u_{l,2}), s_{l,2}, l_{l,2,1}] \ldots$ $[(u_{l,n(l)-1}, Z), s_{l,n(l)}, l_{l,n(l),1}]$<br>where:<br>$u_{l,a}$ is the a<sup></sup>th node traversed by path i<br>$s_{l,b}$ is the bth span traversed by path i<br>$(A, u_{l,a})$ are the nodes terminating span $s_{l,b}$<br>$l_{l,b,c}$ is the cth link used on the bth span traversed by path i<br>$n(l)$ is the logical path length of path i |
| $R_{xy}^{AZ}$ | the set of all $k_{xy}^{AZ}$ restoration paths, i.e. the restoration plan for node pair (A,Z) |
| R | network redundancy |
| Rn | the total fraction of a network's working capacity that is protected |
| $Rn_{wc}$ | the lowest $Rs_i$ of any span in the network |
| $Rs_i$ | the restorability of span i, having $w_i$ working links |
| $s_b$ | the number of spare links on span b |
| S | the number of spans in a network |
| $t_{p1}$ | the time required to complete the first path of a restoration plan |
| $t_{p,i}$ | the individual outage experienced by the ith working path restored |

| | |
|---|---|
| $t_{p,avg}$ | the average of $t_{p,i}$ over all working links restored in a restoration event |
| $t_R$ | the time required to complete the last path of a restoration plan |
| $t_{95}$ | the time required to restore 95% of all individual links |
| $T_{xy}^{AZ}$ | number of link disjoint restoration paths that are topologically feasible between node pair A-Z, combined with those restoration paths from every other demand pair to be restored, over the unused spare links in the network after the failure of span x-y. |
| w | the average demand lost per relation over all span cuts |
| $w_j$ | the number of working links on span j |
| W | the average working capacity per span |
| $X_{xy}^{AZ}$ | the number of demand units lost by demand pair A-Z upon the failure of span x-y, sometimes shortened to $X_i^r$ where r is a tag for demand pair A-Z and i is a tag for span *xy*. |
| $\delta_{i,j}^{r,p}$ | takes the value of 1 if the $p^{th}$ restoration route for demand pair r after the failure of span i uses span j, and 0 otherwise |
| $\delta_{xy,b}^{r,l}$ | takes the value of 1 if the $l^{th}$ restoration path for demand pair r after the failure of span *xy* uses span b, and 0 otherwise |
| $\zeta_j^{r,q}$ | takes the value of 1 if the $q^{th}$ working route for demand pair r uses span j |

# PART I: Introduction and Capacity Placement Methodology

The two main research contributions of this thesis are a path restorable network capacity placement methodology, and an optimized distributed real time solution to the path restoration problem, named OPRA. The first part of this thesis focuses on the capacity placement methodology. It is described in detail in chapter 4. Chapters 1 through 3 define the restoration problem and develop the background necessary to differentiate the current work from prior research.

# Chapter 1. Introduction and Background

## 1.1 The Restoration Problem

"Hurricanes, fires, floods, tornadoes, terrorist bombings: These and other disasters exact a great physical, emotional, and economic toll. The disruption of telecommunication services is part of this toll - services that are vital not only for life under normal circumstances, but also for recovery from disasters themselves." [15]

With the integration of computers and telecommunications, the dawn of the so-called "information age" is upon us, and telecommunications has become a vital part of our life. Service disruption is no longer being tolerated and at the same time, the consequences of service disruption are becoming more severe.

Disrupting telecommunication services causes both tangible and intangible losses for users as well as for service providers. To ensure service continuity, service providers have increased their efforts to alleviate such disruption. A challenging task centered on these efforts is how to ensure service continuity affordably. This is formally called the *restoration problem* in modern telecommunications practice.

Solving the restoration problem involves reestablishing the carrier signals lost due to a failure such as a cable cut. In the transport network, restoration is achieved by rapid and accurate rerouting of carrier signals over a set of replacement paths through the spare transmission capacity in the network. These restoration paths need to be independent from each other as well as capacity and topologically feasible in the network. The restoration problem is significantly different from the well-studied packet routing and call routing problems and presents demanding real time computational challenges. To date there has been no published report of an *optimized distributed real time* solution to the *path* restoration problem. Such a technique and a network capacity design methodology which exploits this technique are the main contributions of the present research.

## 1.2 Economic Impact of Outages

Optical fibre systems are today the transmission medium of choice for telecommunication networks. Their higher capacity, higher reliability, longer repeater spacing, greater security, smaller size, growth potential, and lower cost when compared to copper and radio transmission systems, has led to the large scale deployment of fibre optic transmission systems (FOTS) in the transport network. Furthermore, the large capacity of FOTS, and the fact that they become more economical when loaded with large amounts of aggregated traffic, has resulted in sparsely connected transport networks with fewer but more heavily loaded routes.

Despite the advantages of FOTS, it is a cable-based technology and susceptible to frequent damage. Due to the high volume of traffic being carried by fewer fibre systems, the consequences of a service disruption are severe. With transmission capacities of 2.488 Gbps (OC-48) possible over fibre optic systems, up to 37 152 phone conversations (or an equivalent amount of data) can be simultaneously transmitted over a single pair of optical fibres. The failure of a single cable, which may carry up to 48 fibres, can therefore seriously degrade the performance of a telecommunication network. Furthermore, due to the increased necessity of communications with bankers, purchasing managers, stock brokers, retailers, and so forth, a network failure may result in revenue losses of $75 000 US to $100 000 US per minute of outage [25].

### 1.2.1 Restoration Time Objectives

The motivation for pursuing the fastest possible restoration times is not purely economic. A sudden disconnection during an active transaction in networks of point-of-sale transaction terminals, automated tellers, personal computers, etc. can cause an uncertain state from which the end application may not recover, and if it does, it may be at the expense of terminating the transaction in progress. Therefore, as the volume of data communication grows, it becomes especially important to avoid the loss of connections in progress.

A major connection-dropping event not only threatens user applications but also the stability of circuit switches. It is not known with confidence how software-controlled switching machines react to dynamic traffic surges resulting from major connection-dropping events and the ensuing synchronized call re-attempts. Many suspect that more than one total crash has been due to such transients [25]. Therefore, restoring a failure quickly before the call-dropping threshold mitigates the threat of transients to circuit switches, decreases the possibility of malfunctions in user applications, and minimizes the economic impact on industry.

The call-dropping threshold for most circuit switched services is 2 seconds. Voice service is not significantly degraded if restoration is achieved prior to 2 seconds because only those voiceband circuit-switched calls affected by the failure that are on trunks associated with older channel banks are dropped, and the total number of trunks in the public-switched network that are carried on older-type channel banks is less than 12%. [61] Furthermore, as data rates increase it becomes especially advantageous to restore data services quickly so that the amount of application information which must be recovered is minimized. Figure 1.1 below illustrates the impact of various restoration times on industry and the network. The time objective of the restoration algorithm presented in this thesis dissertation is 2 seconds.

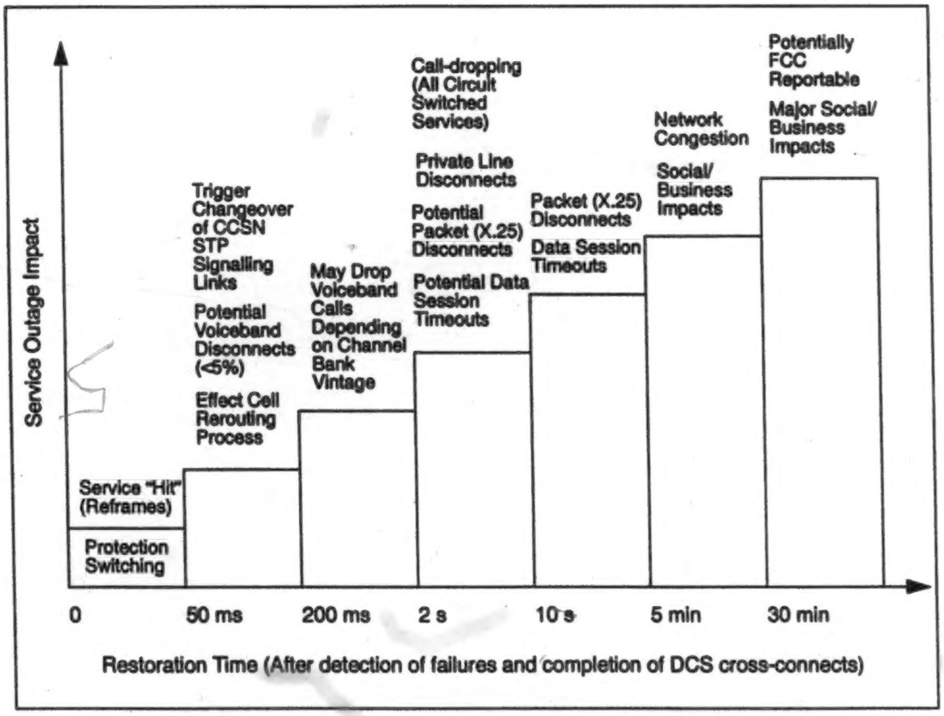


Figure 1.1. Restoration Time Impact [adapted from 61]

### 1.2.2 Capacity Efficiency Objectives

Fast restoration times can not always be achieved economically because fast simplified restoration generally requires increasing network redundancy. Network redundancy is defined as the ratio of spare to working capacity in a network. Networks with large redundancies are less economical because the spare capacity required to protect them is expensive. In addition to minimizing restoration time, minimizing the capacity requirements of the transport network while maintaining the ability to restore the most common types of failures quickly, is an equally important objective when solving the restoration problem. The network design methodology presented here minimizes the capacity requirements of a transport network while maintaining the ability to restore the most common types of failures within two seconds.

It is of course impossible and uneconomical to protect a network from every type of failure. Considering the infrequency of multiple simultaneous cable cuts and the low failure rate of circuit switches, modern transport networks are generally engineered to completely restore only single cable cuts. The most common type of failure considered in the restoration problem is the single cable cut, and unless otherwise specified is implied here when referring to a failure. However, with the restoration method developed in this thesis, recovery from node failures as well as multiple simultaneous cable cuts is possible.

Together, a design methodology which minimizes the capacity requirement of the transport network and a restoration algorithm which restores any single cable cut solves the restoration problem. To restore all lost signals in an efficiently spared network the restoration algorithm must use a minimum amount of spare capacity. Often speed and capacity efficiency must be traded off against each other. The restoration algorithm presented in this thesis aims to use only a portion of the spare capacity required by existing or proposed restoration algorithms and still restore a failure within two seconds.

## 1.3 Present Methods for Transport Network Restoration

Currently diverse routed 1:1 protection switching has been widely deployed to protect the transport network from failures, not so much because it achieves the fastest restoration times possible as shown in Figure 1.1, but because it can be implemented using current technology. Two restoration techniques employing protection switching include Automatic Protection Switching (APS) and Self-Healing Rings (SHR) [16, 22, 69].

### 1.3.1 Automatic Protection Switching

APS is a restoration technique that automatically reroutes signals from a working line to a dedicated protection line during a signal outage. APS is the simplest and fastest facility restoration technique. The APS switching technique is sometimes referred to as span switching because APS is performed on an entire optical system.

A span is defined as the collection of all point-to-point DS-3 or OC-n links, working or spare, in parallel between two nodes. A span may be thought of as a pipe through which traffic can "flow", although its composition as discrete links is ultimately important because a solution to the restoration problem must specify precise connections between links.

When the protection system is physically diverse from the working fibre sheath (referred to as Diverse Protection or DP), APS can protect a transport network from single cable cuts. However, only when every working system is paired with a protection system, referred to as a 1:1 APS architecture, is 100% restoration possible. The minimum redundancy of a survivable transport network employing 1:1/DP APS systems is therefore 100%.

Given that a 1:1 APS architecture does not share a protection system with multiple working systems, the 1:1 APS switching protocol defined in the Synchronous Optical Network (SONET) standards permanently bridges the head end and only switches at the tail end as shown in Figure 1.2. Referring to Figure 1.2, the 1:1 APS switching protocol can be summarized as follows:

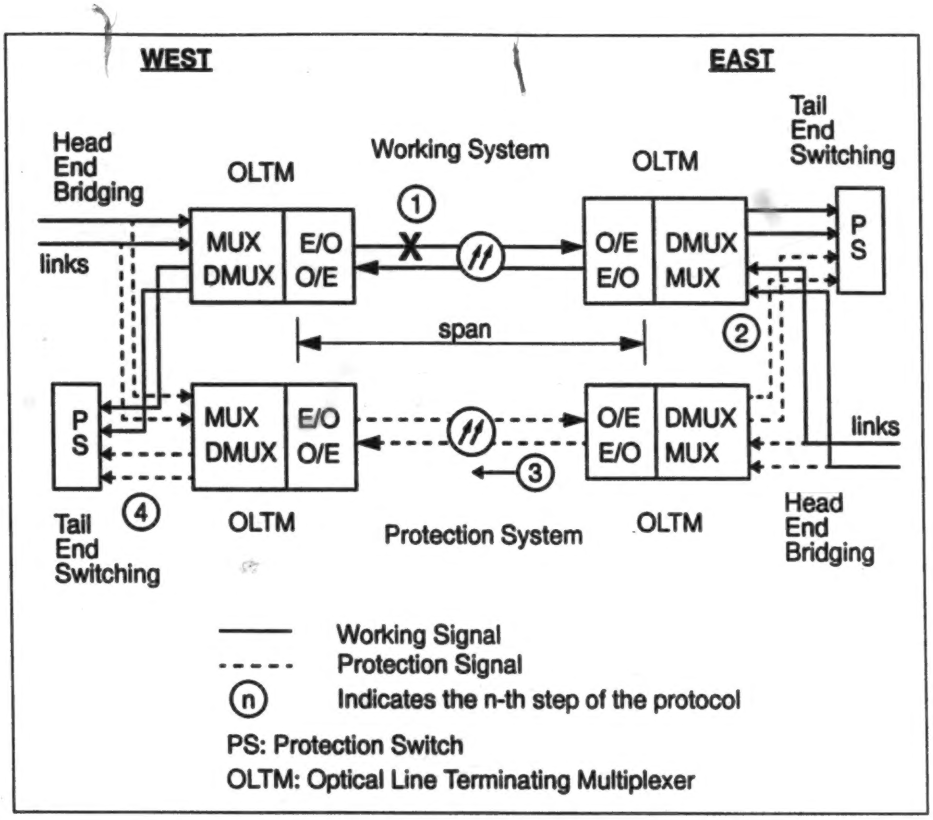

WEST
EAST
Head End Bridging
links
OLTM
MUX DMUX
E/O O/E
Working System
①
X
OLTM
O/E E/O
DMUX MUX
Tail End Switching
P S
span
②
P S
MUX DMUX
E/O O/E
O/E E/O
DMUX MUX
links
③
④
Tail End Switching
OLTM
Protection System
OLTM
Head End Bridging
Working Signal
Protection Signal
ⓝ Indicates the n-th step of the protocol
PS: Protection Switch
OLTM: Optical Line Terminating Multiplexer

Figure 1.2. 1:1 Automatic Protection Switching [adapted from 69]

1. A uni-directional failure of multiple links is recognized by the receiving performance monitor at the east end of the span.

2. The APS controller at the east end of the span operates a tail-end transfer relay in the protection switch and connects to the working links being received over the protection system.

3. The APS controller at the east end informs the west end controller of the failed links

4. The APS controller at the west end of the span operates a tail-end transfer relay in the protection switch and connects to the working links being received over the protection system.

In the event of a bi-directional failure, both ends of the span may simultaneously bridge to the protection system. Note that the protocol only communicates over the protection system.

### 1.3.2 Self-Healing Rings

Networks employing 1:1/DP APS can be evolved to Self-Healing Rings (SHRs). SHRs may require less fibre, fewer optical/electronic devices, and fewer regenerators than 1:1/DP APS systems. The evolution to a ring architecture is generally not difficult considering the numerous similarities between APS systems and SHRs.

APS systems are line switched systems and are most closely related to Bidirectional Line Switched Self-Healing Rings (BLSRs). However, all SHR networks provide redundant bandwidth and/or network equipment so disrupted services can be automatically restored following network failures like APS systems. A SHR network chains multiple Add Drop Multiplexers (ADMs) together in the form of a working ring and establishes a second spare communications path parallel to the first as shown in Figure 1.3. ADMs add traffic to, drop traffic from, and pass traffic along the ring, as well as performing protection switching in the event of a failure. Survivable ring networks restore a failure by employing protection switching between the spare and working ring similar to the operation of 1:1/DP APS architectures. SHR networks also require a minimum of 100% redundancy because there must be sufficient restoration capacity on the spare ring to restore any failure on the working ring, much like 1:1 APS systems which require one-to-one redundancy between working and protection systems.

Numerous SHR architectures are possible, each of which offer complete survivability for single cable cuts. If a ring performs protection switching between the ADMs terminating a failure, the architecture is referred to as a line or span switched ring. If protection switching is performed between the ADMs which add traffic to and drop traffic from the ring, the architecture is referred to as a path switched ring. When the spare and working rings only transmit in one direction, the ring is referred to as unidirectional, otherwise it is referred to as bidirectional. Furthermore, at the physical level, the spare and working rings associated with a bidirectional architecture may or may not share the bandwidth of a single FOTS system, requiring that bidirectional rings either be designated as 2 fibre (spare and working rings share the bandwidth of one FOTS system) or 4 fibre (spare and working rings reside on different FOTS systems). The four different logical ring design architectures are summarized in Figure 1.3.

Like APS systems, SHR networks can restore a network very quickly. SHR networks typically achieve restoration within 150 msec. However, these low restoration times are achieved at the expense of high network redundancy. Each working ring in a SHR network requires an entire spare ring to recover from a failure. Therefore, the redundancy of a SHR network can never be less than100%, and is typically around 300% given that the last rings placed in a multiple ring design often protect a small amount of working capacity. [27, 30]

To completely protect a network using ring technology, multiple rings are required. Initially the design must define the rings (number, size, and location) and route the working traffic through these rings from source to destination. Then the capacity required by the spare rings may be calculated. Growing a multiple ring network may be even more difficult than engineering the initial design. The addition of a new ADM to a SHR network requires at best breaking a single ring and inserting the node and adding a new ring requires at best over-laying the new ring on an existing design. At worst adding a new ADM or ring requires redesigning the entire SHR network.

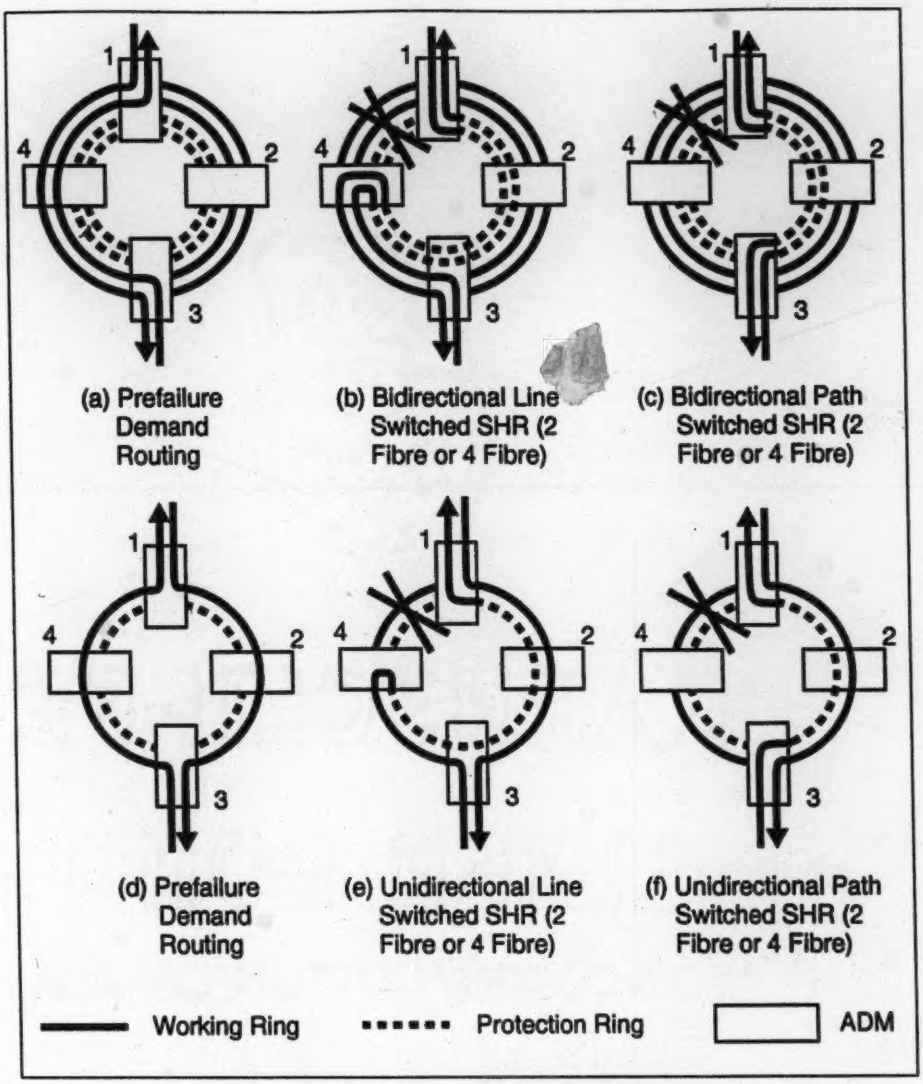

1
2
3
4
(a) Prefailure Demand Routing
(b) Bidirectional Line Switched SHR (2 Fibre or 4 Fibre)
(c) Bidirectional Path Switched SHR (2 Fibre or 4 Fibre)
(d) Prefailure Demand Routing
(e) Unidirectional Line Switched SHR (2 Fibre or 4 Fibre)
(f) Unidirectional Path Switched SHR (2 Fibre or 4 Fibre)
Working Ring
Protection Ring
ADM

Figure 1.3. Types of Self-Healing Rings

## 1.4 Mesh Restoration of Transport Networks

SHR and APS architectures can be implemented using current technology. However, the recent development of Digital Crossconnect Systems (DCS) and the adoption of networks with physically diverse routes and closed topologies has created the opportunity to advance the state of network restoration.

DCS machines are in many respects similar to a computer, and a transport network consisting of Digital Crossconnects is similar in many respects to a distributed multiprocessor. Mesh-based survivable or restorable network architectures exploit the intelligence of a DCS-based transport network to minimize the amount of spare capacity required to protect working demands [2, 7, 11, 13, 28, 29, 57, 71]. In mesh restorable networks the spare capacity on one span contributes in general to the survivability of many other spans. Network redundancy is not dedicated to restoration of one span or ring. These networks are called "mesh restorable" not to imply that the network is a full mesh, but to reflect the ability of the rerouting mechanism to exploit a mesh-like topology through highly diverse and efficient rerouting of failed signal units.

Mesh restoration typically is envisioned to operate at the logical layer rather than the physical layer. Physical layer network designs like APS and SHR are sectionalized and use separate fibre facilities and/or equipment as protection resources. Logical layer designs are more global in the sense that they share the spare capacity on all spans to protect the entire network. By sharing spare capacity, mesh restoration requires significantly less total network capacity than APS or SHR to fully protect a network. Mesh techniques can yield full survivability with redundancy that is inversely proportional to the average nodal degree of the network. For real networks which tend to exhibit nodal degrees between 3 and 4.5 there is the prospect of full survivability with as little as 30% to 50% redundancy. [27]

Furthermore, it is generally acknowledged that mesh restorable networks can grow more elegantly and flexibly than ring or APS restorable networks. The addition of a new switch to a mesh network simply involves informing any node to which the new switch is connected what the spare capacity connecting them is, and ensuring 100% network survivability involves at most increasing the spare capacity on a few spans.

Considering the capacity savings of mesh-restorable networks, the relative simplicity of their planning, and their associated cost savings which will grow as switching costs decrease [27], mesh restorable networks appear to be an attractive alternative technology to APS and SHR. However, the advantages of mesh restoration are gained at the expense of increased complexity. Whereas the main difficulty with APS and SHR networks is high cost and capacity consumption, the challenge in a mesh design is minimizing the time it takes to restore a failure. Minimizing the restoration time often leads to an increase in system complexity.

A variety of mesh restoration techniques of varying complexity exist. The following sections explain, categorize and differentiate these techniques. The new restoration algorithm presented in this thesis is also categorized and distinguished from previously proposed restoration mechanisms.

### 1.4.1 Centralized vs. Distributed Restoration

At the most general level of abstraction, mesh restoration requires three conceptual steps: (a) accessing a network description, (b) computing a re-routing plan, and (c) deploying cross-connection actions to put the plan into effect. Centralized and distributed restoration can be differentiated by examining the steps of the restoration process.

In step one of the restoration process, centralized mesh restoration accesses a database at a central controller which stores information about all network nodes, connectivity maps, and spare facilities. In distributed restoration the network itself is the database; rather than accessing a central controller, each DCS obtains local network information from the links impinging on it.

To fulfil step two of the restoration process, centralized restoration computes the best rerouting paths for all failed signals based on the most recent network information available in the controller's database. Distributed restoration computes the rerouting plan in a distributed fashion across the entire network so that each DCS only computes the part of the composite routing strategy which it is required to implement. The computed set of restoration paths form the rerouting plan in both cases.

In step three of the restoration process, centralized restoration requires downloading the rerouting plan to all DCS machines. However, distributed restoration leaves the computed set of restoration paths in place at each DCS node, obviating the need to download any rerouting plan.

While centralized and distributed restoration disperse information of the rerouting plan differently in step three of the restoration process, both centralized and distributed restoration may deploy the cross-connects required to implement a rerouting plan in the same way. The ways in which cross-connection actions may be deployed at a DCS are explained in section 1.4.3.

Centralized restoration is challenged with problems related to the size, cost, complexity, and vulnerability of the surveillance and control centre needed for transport management. A centralized system is also dependent on the ability to maintain a complete, consistent, and accurate database image of the network which necessitates redundant high-availability telemetry arrangements. As a result, centralized restoration is not only slower in real time than distributed restoration but runs the risk that a failure in the network will coincide with downtime at the central control site or a failure in the telemetry arrangement.

In a distributed approach there are no dependencies on telemetry or a central control site; the network is the computer on which the reconfiguration algorithm runs. Therefore, the distributed approach is less vulnerable than a centralized approach. Furthermore, distributed mesh restoration algorithms have the potential to compute a rerouting plan much faster than centralized algorithms because they use the network as their database, and perform distributed processing over all DCSs. However, distributed restoration algorithms tend to be more complex than centralized algorithms because they must ensure that the routing decisions computed in isolation at every DCS coordinate with the routing actions taken by all other nodes. The restoration algorithm presented in this thesis is a distributed mesh restoration technique.

### 1.4.2 Span vs. Path Restoration

In addition to categorizing a mesh restoration technique as either centralized or distributed, mesh restoration can be divided further into two classes: path and span restoration. Span restoration re-routes failed signal units over a set of replacement paths through the spare capacity of a network between the two nodes terminating a span cut.

Path restoration reroutes failed signal units over a set of replacement paths between each of the end source and destination nodes affected by a failure. Figure 1.4 illustrates the difference. When span S-T fails, span restoration finds replacement path segments directly between nodes S and T, whereas path restoration finds end-to-end replacement paths between demand pairs A-C and B-D.

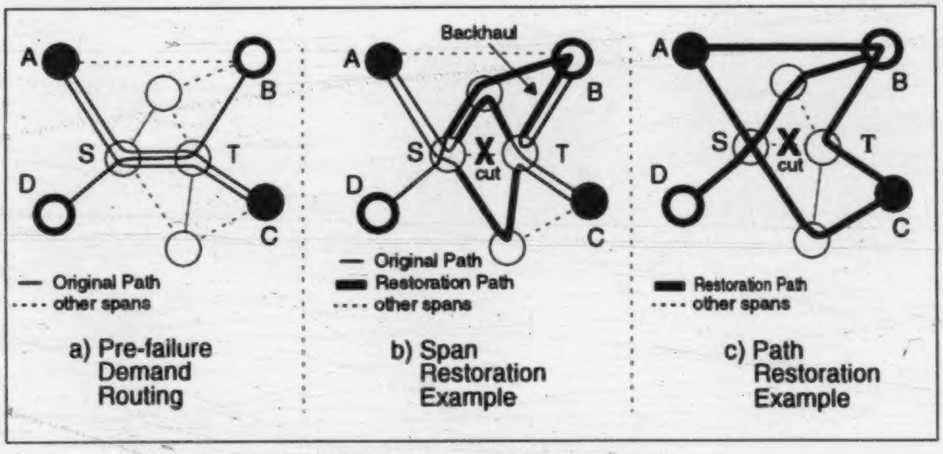

A
B
S
T
D
C
— Original Path
···· other spans
a) Pre-failure Demand Routing
Backhaul
A
B
S
X
cut
T
D
C
— Original Path
▬ Restoration Path
···· other spans
b) Span Restoration Example
A
B
S
X
cut
T
D
C
▬ Restoration Path
···· other spans
c) Path Restoration Example

Figure 1.4. Span vs. Path Restoration

In principle path restoration is more capacity efficient than span restoration because it distributes simultaneously required replacement paths over a wider region of the network, increasing the alternatives available when optimizing capacity placement such that a reduction in spare capacity is possible. In contrast, span restoration creates replacement paths between the end-nodes of a failure so that inefficiencies may result in the form of "backhauls" as shown in Figure 1.4. Backhauls result in a greater use of spare capacity than is required to restore a failure. They arise in span restoration and not in path restoration because span restoration does not consider the end-to-end routing of the actual demands affected by a span cut. Path restoration attempts to quickly and efficiently re-provision all of the demands affected by a failure across the entire network.

Path restoration also inherently provides for restoration of transit paths after node loss. Given that 100% restoration will likely be impossible after a node failure because those connections terminated on the failed node cannot be recovered, it may be undesirable for a path restoration algorithm to simply result in a maximum of total capacity restored over all relations affected, or accept a random pattern of replacement paths. Rather, node recovery may require an overall pattern of individual restoration levels on each relation that is proportional to its prefailure connectivity [47]. Span restoration algorithms do not aim to address node failures.

The elimination of backhauls and the ability to restore node failures suggests that it will be desirable to use path restoration rather than span restoration. However, path restoration is more complex than span restoration because it requires finding a set of replacement paths between multiple source and destination pairs, as opposed to rerouting the same total number of failed signal units between a single pair of nodes. While span restoration algorithms which maximize network restorability and use as little spare capacity as possible find a set of k-successively shortest link disjoint replacement paths between the end nodes of a span cut [19], it is not known how the replacement paths between the source and destination of all working connections affected by a failure should be routed to achieve the max-flow. The restoration algorithm presented here implements path restoration and uses a heuristic to find a set of link disjoint replacement paths which results will show very often achieves the max-flow possible between those relations affected by a failure.

### 1.4.3 Preplanned vs. Dynamic Restoration

In mesh restorable networks, cross-connections can be deployed at a DCS either by referencing a database of restoration paths, known as preplanning, or the cross-connections can be derived dynamically as a restoration algorithm identifies rerouting paths. The preplanned method requires that each DCS store all, or most, of the rerouting information related to the restructuring of the network for all preplanned failure scenarios. In the event of a failure, techniques using preplans, such as those described in [12, 42, 58], selectively broadcast information regarding the failure throughout the network. Each DCS receiving one or more of these messages determines the cross-connections it needs to implement in order to restore the failure based on the rerouting information stored in its local database.

In contrast, the dynamic method does not require the DCS to store any database of network configuration. Rerouting decisions are made in real time based on either the state of the network at the time of the failure (as is the case in distributed restoration), or the state of the network stored in a database at a central controller (as is the case in centralized restoration), and cross-connections deployed as restoration paths are identified. Figure 1.5 summarizes the architectural differences between distributed and centralized, and preplanned and dynamic mesh restoration. Not shown in Figure 1.5 is the central operations and management centre (OAM) common to all architectures which collects alarms, and ratifies the completed restoration plan of any restoration mechanism as an follow up action.

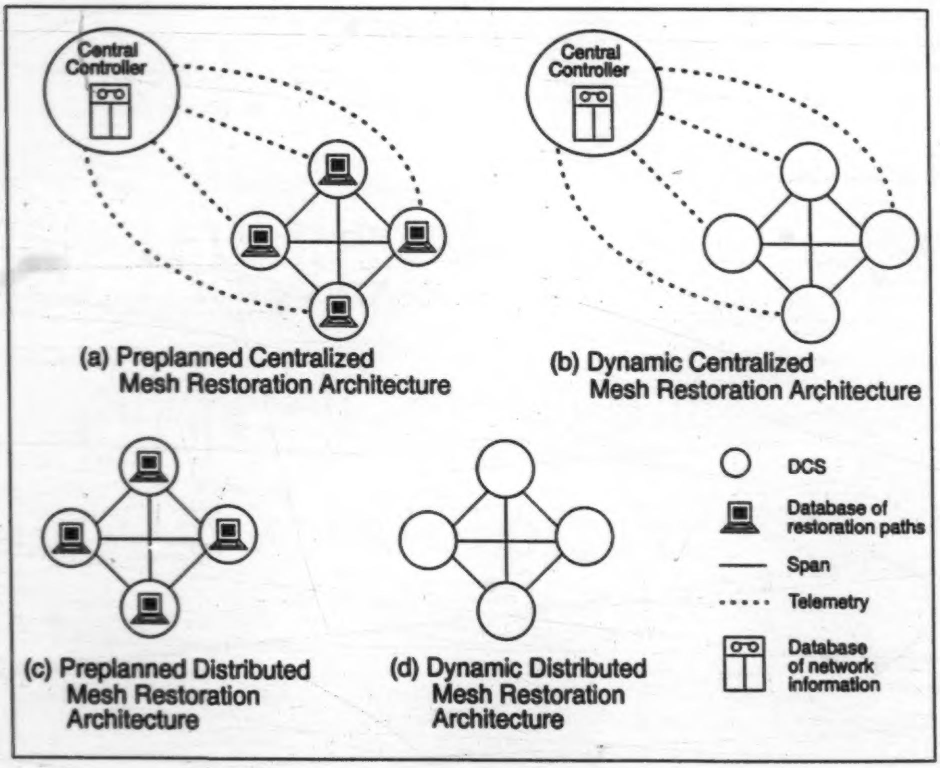

Central Controller
Central Controller
(a) Preplanned Centralized Mesh Restoration Architecture
(b) Dynamic Centralized Mesh Restoration Architecture
(c) Preplanned Distributed Mesh Restoration Architecture
(d) Dynamic Distributed Mesh Restoration Architecture
DCS
Database of restoration paths
Span
Telemetry
Database of network information

Figure 1.5. Mesh Restoration Architectures [adapted from 69]

In the event of a failure, centralized dynamic mesh restoration initiates a search for the best rerouting path for each failed signal based on the current network state stored in the controller's database. After finding the optimal restoration paths[1], the controller sends commands to selected DCSs prompting them to change their switching matrixes and reroute the failed signals. Modern centralized dynamic restoration techniques such as those reported in [7, 8, 14] are capable of restoring a network failure within minutes, but unable to achieve restoration within seconds because of the time required to collect alarms and download the new routing plan.

In distributed dynamic mesh restoration, the network is the database that is accessed at the time of a failure and used to compute the optimal restoration pathset. Completion of the restoration algorithm leaves a restoration plan in place exactly as would have been downloaded had the plan been centrally computed.

In general, compared to the dynamic rerouting method, the preplanned method requires higher memory, may have more difficulty adapting to rapid network changes, and may have a lower reliability because it cannot prepare and place preplan information in all nodes for all possible failure scenarios. However, the preplanned method can be faster because it eliminates any real-time dependencies on a central controller and minimizes the processing requirements at a DCS during a failure. Preplanned restoration opens up the possibility that a fast cross-connection DCS could actually match survivable protection switching architectures in terms of restoration speed, while still deploying capacity efficient restoration pathsets which are the main benefit of mesh restorable networks. [28]

Any span or path, centralized or distributed, mesh restoration algorithm can deploy the required cross-connections by either referencing a preplanned database or dynamically. The distributed path restoration algorithm presented in this thesis restores failures by deploying cross-connections dynamically.

### 1.4.4 Optimized Distributed Dynamic Path Restoration

Ideally a restoration algorithm should restore a network failure quickly (within two seconds), require little administration overhead, handle numerous failure scenarios (not

---

1. Optimal restoration paths maximize the restorability of a network, Rn as defined in chapter 5, while using as little spare capacity as possible as explained in chapter 3.

just single span cuts), be highly reliable, easily accommodate network growth, adapt itself to any network topology, and require a minimum amount of spare capacity. Of all the restoration architectures discussed thus far (see Figure 1.6), mesh restorable networks employing distributed dynamic path restoration have the greatest potential to satisfy these goals. One of the main contributions of the present research is a distributed dynamic path restoration algorithm, named OPRA, that satisfies all of these goals.

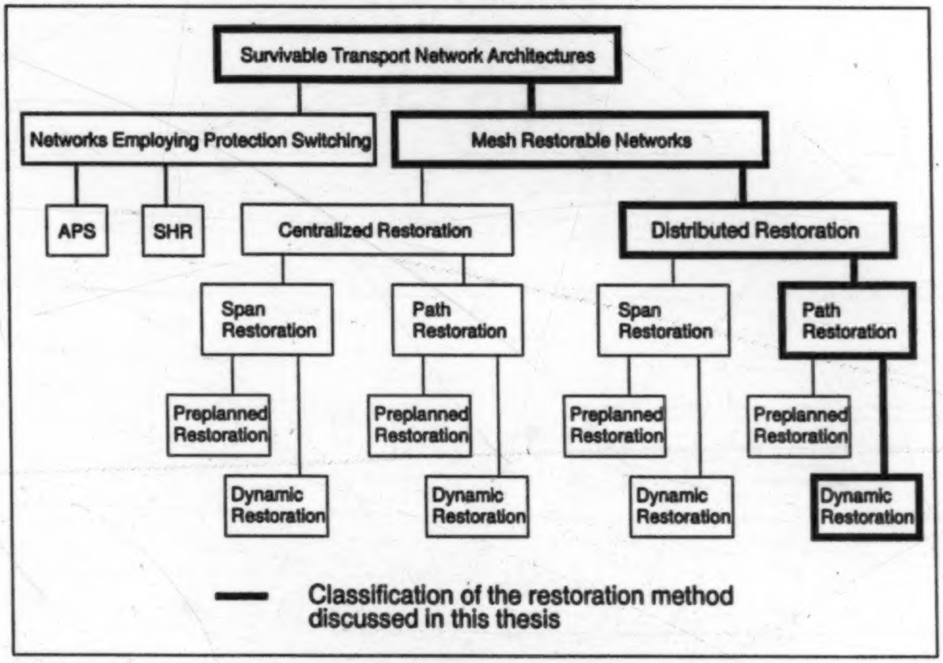
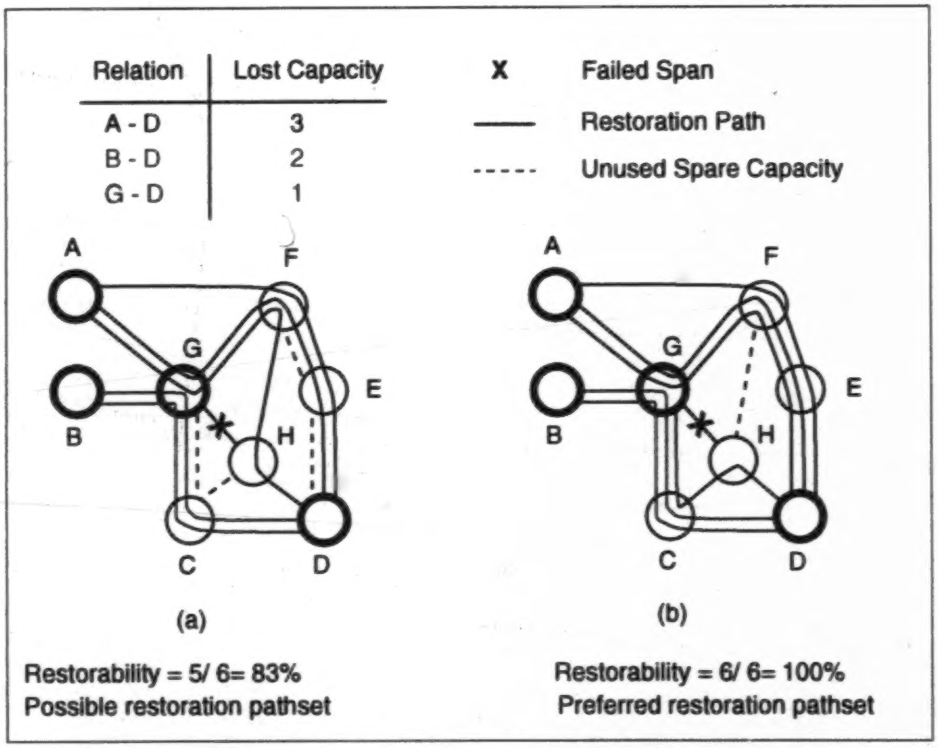

Survivable Transport Network Architectures
Networks Employing Protection Switching
Mesh Restorable Networks
APS
SHR
Centralized Restoration
Distributed Restoration
Span Restoration
Path Restoration
Span Restoration
Path Restoration
Preplanned Restoration
Dynamic Restoration
Preplanned Restoration
Dynamic Restoration
Preplanned Restoration
Dynamic Restoration
Preplanned Restoration
Dynamic Restoration
Classification of the restoration method discussed in this thesis

Figure 1.6. Survivable Transport Network Architectures [69]

To date several distributed dynamic span restoration algorithms have been reported [2, 7, 11, 13, 29, 57, 71], many of which claim the ability to perform path restoration [11, 43, 55, 57]. In general, any span restoration algorithm can be turned into a rudimentary path restoration scheme by iteratively applying the span restoration algorithm to all affected source - destination demand pairs. Using a span restoration algorithm to perform path restoration in this way was previously called Capacity Scavenging [28]. While Capacity Scavenging can be used in path (and hence node) restoration, the recovery patterns obtained from uncoordinated concurrent (or arbitrary sequential) execution of a span restoration algorithm for every demand pair affected by a node or span failure may not yield an optimal allocation of recovery levels amongst affected demand pairs.

The sub-optimality of "ad-hoc" path restoration that may result from Capacity Scavenging is illustrated in Figure 1.7. Capacity Scavenging is just as likely to find the first pathset identified in Figure 1.7 rather than the second preferred pathset. As a result, Capacity Scavenging may result in one demand pair receiving a maximum number of restoration paths (relation A-D in Figure 1.7 (a)) and others receiving none (relation G-D in Figure 1.7 (a)). Furthermore, given a network with a fixed amount of spare capacity, Capacity Scavenging might use long restoration paths when a larger number of efficient shorter restoration paths that restore more lost capacity are possible.



| Relation | Lost Capacity |
|---|---|
| A - D | 3 |
| B - D | 2 |
| G - D | 1 |

Restorability = 5/ 6= 83%
Possible restoration pathset

Restorability = 6/ 6= 100%
Preferred restoration pathset

Figure 1.7. Example illustrating sub-optimality of "ad-hoc" path restoration

While a few distributed dynamic path restoration algorithms have been reported [42, 62], none attempt to find the optimal restoration pathset. The term optimal as used here implies a pathset which restores the maximum amount of lost demand topologically feasible. Distributed dynamic path restoration algorithms developed to date have focused on achieving restoration within the two second call-dropping threshold, and have given capacity efficiency secondary consideration. The research presented here is unique in that it is the first distributed dynamic path restoration algorithm which attempts to configure the surviving spare links of a path restorable network into an optimal multicommodity max-flow pathset, and do so within the two second call-dropping threshold.

## 1.5 Capacity Placement for Mesh Survivable Networks

Solving the restoration problem not only requires restoring a failure but determining an economic capacity placement which permits full restoration. While sections 1.3 and 1.4 have introduced and discussed prior work regarding the development of restoration algorithms, this section categorizes the capacity placement technique presented in this thesis and differentiates it from previously proposed capacity placement methods.

### 1.5.1 Using Heuristics to Solve the Capacity Placement Problem

Two general techniques have been used to solve the complex capacity planning problem. They include using either integer programming or heuristic techniques. Heuristics are usually used to guide a greedy algorithm that incrementally adds or deletes spare capacity from a network, as in [11] and [67]. For example, the algorithm in [67] realizes a near-minimum spare link assignment in a span restorable network in two phases: forward synthesis and design tightening.

Phase one in [67] iteratively addresses the question: given one new spare link to 'spend', where should it be placed to yield the greatest step increase in aggregate network restorability? Restorability is defined formally later in this thesis, but may be interpreted here as the ratio of restored to lost working capacity. In the forward synthesis phase, a spare link is temporarily added to a span 'X'. Then, each span in the resulting network is cut one at a time in sequence, and the working capacities of the cut span rerouted through the spare capacity of the surviving portion of the network using an efficient k-shortest link disjoint paths algorithm. The average restorability of all span cuts is then calculated, and the spare link removed from span 'X'. A spare is then added to another span, and the global restorability increase recomputed. After the global restorability has been calculated for all possible placements of the spare, an additional spare is permanently added to the span on which it gives the greatest global restorability increase. This process is repeated, adding one spare at a time, until there is no span on which an extra link gives an increase in network restorability.

At this point, restorability has reached 100% or a stalling point has occurred. A stalling point is a network state in which the addition of a single spare doesn't increase restorability. If restorability is 100%, the initial network spare capacity placement design is complete. If a stalling point has occurred, all possible additions of a two-link path segment are analysed for the greatest increase in global restorability. The pair of link placements which result in the largest increase in restorability are then accepted and the algorithm returns to trying to add one spare at a time. If pairs of links cannot overcome the stalling point, a shortest complete restoration path is added to a span that is currently not fully restorable, and then the algorithm returns to trying to add one spare at a time.

Phase one, forward synthesis seeks a steepest ascent in restorability against redundancy. Tightening in [67] whittles away at redundancy while clamping restorability at 100%. Forward synthesis uses a greedy strategy, so there will generally be some opportunities for improvement by selective eliminations and redistribution of spare capacity. In particular, forward synthesis cannot foresee future additions of spare capacity that may make a present increment unnecessary. Tightening compensates for this.

The first stage of the tightening phase eliminates any spares which can be removed without any reduction in restorability. When no links can be removed without a loss in restorability, tightening examines all combinations of capacity-saving redistribution in the spare link assignments. Tightening searches for opportunities to add "n" spares while removing "n+1" other spares. Redistribution with n=1 is attempted first, then n=2. Tightening is complete when redistribution at n=2 cannot remove any spares without decreasing restorability. There is no conceptual reason to stop at n=2 redistribution, but there is little or no improvement after n=2 [67].

This sort of a capacity placement technique assumes some knowledge of the routing of replacement paths after a network failure. In [67] an efficient k-shortest link disjoint paths (ksp) algorithm is repeatedly used to identify replacement paths and calculate restorabilities. While near optimal *span* restoration algorithms find a set of k-successively shortest link disjoint replacement paths between the end nodes of a span cut [19], it is not known *in general* how the surviving spare links of a *path* restorable network should be configured to achieve the multicommodity max-flow between those relations affected by a failure. A fast centralized algorithm similar to the one used in [67] to calculate restorabilities in a span restorable network has not been identified for path restoration. It is the intent of the present research to compare the spare capacity required in span *and path* restorable networks in which working and spare capacity has been optimized. Without a fast centralized ksp-like algorithm to identify the multicommodity max-flow through the spare capacity of a network between those relations affected by a failure, it is difficult to solve the capacity placement problem in a path restorable network using heuristics. Therefore this thesis solves the capacity placement problem using integer programming techniques.

### 1.5.2 Using Integer Programs to Solve the Capacity Placement Problem

Recently, Linear Programming (LP) or, more specifically, linear Integer Programming (IP), has been applied to the problem of optimal[1] spare capacity planning in a span restorable network. Integer programming requires defining a set of constraints and an objective function which in this case solve the capacity planning problem. However, the number of constraints required by an integer program to find the optimal capacity placement in a network of practical size is often prohibitively large. To function effectively, the constraint system for formulating the IP must be defined carefully.

One approach has been to develop constraints for the spare capacity placement problem based on a network's cutsets after the failure of a span [57, 67]. A cutset is a set of spans which when severed divide a connected network into two distinct parts. Figure

---

[1] The term optimal when used in conjunction with capacity planning, rather than restoration, refers to that network design which has the minimum amount of spare capacity required to restore all individual span failures.

1. 8 shows two cutsets in a small study network. Cutset number 1 consists of spans 1-2, 1-3, 3-5, and 4-5, and cutset number 2 consists of spans 2-3, 1-3, 3-5, and 3-4. Those cutsets which limit the maximum flow possible between those nodes seeking recovery paths after a failure determine the spare capacity required on each of the spans in the network. For example, if span 3-5 fails in Figure 1.8 and the network is fully span restorable, the number of recovery paths possible between nodes 3 and 5 depends in part on the spare capacity on spans 1-2, 1-3, and 4-5 in cutset number 1, and the spare capacity on spans 2-3, 1-3, and 3-4 in cutset number 2. If the spare capacity on each of the spans in cutset number 2 limits the maximum number of replacement paths, or flow, possible between nodes 3 and 5, cutset number 2 is the minimum cutset.

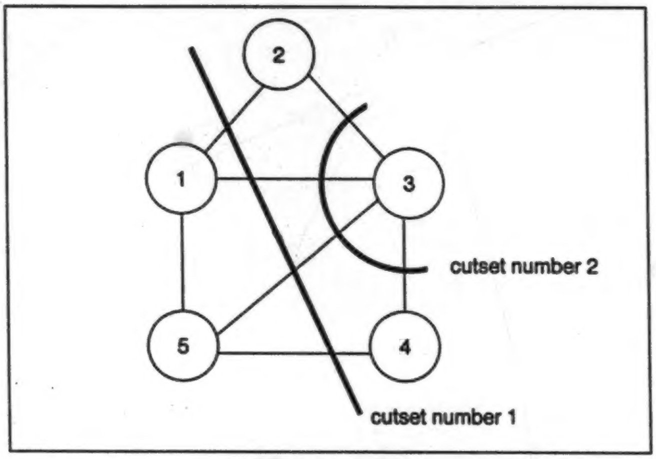


Figure 1.8. Cutsets

In a 100% span restorable network the minimum cutsets for any given span failure must not limit the number of restoration paths between the nodes immediately terminating the failed span to less then the number of working paths traversing the failed span. An IP formulation based on cutsets which finds the optimal spare capacity design limits the minimum flow across all cutsets for all span failures to the lost working capacity on the severed span, and has an objective function that minimize the spare capacity of the network. This type of a constraint formulation is based on the min-cut max-flow theorem [37].

However, it is usually impractical to include all cutsets in the constraint set since the number of cutsets is an exponential function of the size of the network. Choosing cutsets to populate the constraint set iteratively after intermediate, not fully restorable, spare capacity designs, can reduce the size of the constraint set dramatically. Prior work in [57] and [67] populated the constraint sets with cutsets iteratively until the solution provided a spare capacity placement which was 100% span restorable and minimized total spare capacity.

In [67] the set of incident cutsets for every span form the initial constraint set. The incident cutsets are those which contain the spans adjacent to one of the end-nodes of the span under consideration. For example, cutset number 2 in Figure 1.8 is an incident cutset given the failure of span 3-5. If the network is not fully restorable after solving the constraint formulation resulting from all incident cutsets, one restorability constraint is added for each span which is not fully restorable. This additional constraint is found by removing the available restoration paths for that span, and finding a cutset in the resulting network. In [67] this cutset is found by identifying the connected sub-networks in the reduced network, combining sub-networks until only two remain, each containing one of the end-nodes, and then identifying the spans which are adjacent to both sub-networks. These spans compose the new cutset constraint. The IP is then re-executed with the augmented constraint system. After a few iterations, the network typically becomes fully restorable.

A more recent approach has been to specify flow constraints based on a suitable set of predefined routes over which pathsets must be implemented [36,53]. An example of three predefined routes is shown in Figure 1.9. In the event span 3-5 fails, restoration paths would be limited to following one or more of the three routes shown between nodes 3 and 5. An IP formulation using this approach which optimizes the placement of only spare capacity in a 100% span restorable network was reported in [36]. In such a formulation the working capacity design is given and fixed, and the constraint set is based on eligible restoration routes between each pair of nodes terminating a span. When the IP completes, the total flow feasible along those restoration routes is adequate to restore the lost capacity of any span cut.

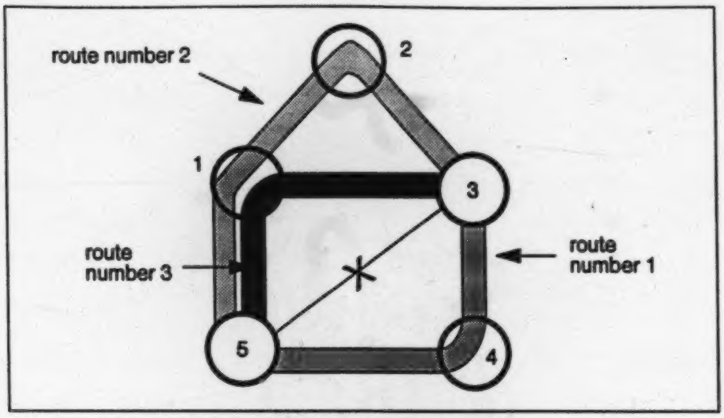


Figure 1.9. Predefined Routes

An IP which jointly optimizes working and spare capacity in the context of a restorable ATM network was reported in [53]. In such a formulation the constraint set is based on eligible working and restoration routes.

The capacity placement technique presented in this thesis also uses flow constraints. In this thesis the spare capacity placement, as well as the joint working and spare capacity placement, of a path restorable network is optimized using an IP with constraints derived from a set of predefined routes. However, unlike the IPs presented in [36] and [53], the IP formulation presented in this thesis uses a highly diverse route set that exploits mesh-like topologies in restorable SDH networks to optimize working and spare capacity, or only spare capacity, in either span or path mesh restorable networks.

Comparing an IP based on route flow assignments to an IP based on cutsets, unlike the constraint set formed using cutsets, the constraint set formed using eligible routes only has to be defined once, and a solution will be ensured in a single linear or integer program run with no iteration. Moreover, while either approach specifies the minimum spare capacity requirements per span, the route-based approach also yields details of the actual paths and routing used to restore each span failure. This information is helpful when evaluating the performance of a distributed restoration mechanism operating in the target capacity design [29], and useful in a centralized restoration mechanism that accesses a database of feasible restoration paths [7].

## 1.6 Research Contributions of this Thesis

Two of the main research contributions of this thesis are a path restorable network capacity design methodology using integer programming, and an optimized distributed real time solution to the path restoration problem, named OPRA. Together these advances make path restorable transport networks feasible. In the envisaged network, capacity would be minimized using the IP design methodology presented, and failures restored in real time in an entirely autonomous distributed manner using OPRA.

The integer program presented in this thesis is the first precise formulation published which can jointly specify the working and spare capacity design of a synchronous digital path restorable transport network, and is flexible enough to accommodate different capacity design options such as stub release. (Stub release is explained in chapter 4.) In addition, this IP specifies the optimal routing of working and restoration paths. Although this pathset information is used to evaluate the performance of OPRA in this thesis, it merits mentioning as a research contribution because it is possible to use this restoration pathset information to achieve centralized restoration.

OPRA is the result of the first thorough exploration, development, realization, and testing of a bidirectional distributed path restoration algorithm. While OPRA defines one particular method for restoring network failures, two central principles integral to OPRA discussed in detail in chapter 6 that are applicable to any path DRA and represent significant research contributions include the interference heuristic and the end-node bottleneck traversal problem.

Furthermore, the conventional program written to test the interference heuristic named Interference Tester represents original work, and merits mentioning as a research contribution because, like the IP, it identifies a restoration pathset that could be used to achieve centralized restoration.

Another research contribution stemming from the investigations of a distributed path restoration algorithm is the realization of a general purpose testbed environment for continued DRA research and/or development. This testbed is an experimental system rather than just a simulator which permits direct transfer of a distributed restoration algorithm's code if the logical node (DCS) model defined in the testbed is complied with.

The research contributions discussed above are summarized in the following list:

1. OPRA
2. the integer program formulation for optimal[1] capacity design
3. the interference heuristic
4. recognizing and solving the end-node bottleneck traversal problem
5. Interference Tester
6. the pathset identified by the integer program formulation
7. a general purpose DRA testbed environment

## 1.7 Outline of Thesis

The first (present) chapter introduced the restoration problem and presented the background necessary to differentiate the current research from prior work.

Chapter two reviews transport networks and the technology used to facilitate mesh restoration. In this review the essential differences between SONET and ATM, and how those differences effect the functioning of a mesh restoration algorithm are highlighted.

Chapter 3 formally states the path restoration problem using graph theoretical terms and explains the differences between it and the packet routing and call routing problems.

Chapter 4 describes a method for capacity optimization of path restorable networks and quantifies the capacity benefits of path restoration over span restoration. The further benefits of jointly optimizing working and spare capacity placement in path restorable networks are also quantified.

Chapter 5 is devoted to consideration of appropriate figures of merit through which the performance and efficiency of a new path restoration algorithm (here OPRA) can be assessed systematically and quantitatively. Collectively chapters 1 through 5 establish the framework necessary to describe OPRA in detail.

---

[1] The term optimal when used in conjunction with capacity planning, rather than restoration, refers to that network design which has the minimum amount of spare capacity required to restore all individual span failures.

Chapter 6 begins to explain the proposed distributed dynamic path restoration algorithm. The principle at the heart of OPRA which enables it to find a near optimal set of link disjoint replacement paths within two seconds is introduced.

Chapter 7 explains how OPRA uses state-based signalling as the framework for interaction between nodes, and the logical environment within which OPRA functions.

Chapter 8 presents a detailed description of OPRA emphasizing the logic necessary to restore a failure. The actions each DCS in a network must perform in response to various interrupts received during the restoration process are explained in this chapter.

Chapter 9 presents the procedures required to effectively test OPRA using discrete event simulations. The nodal logic presented in this chapter supplement OPRA's core procedures presented in the previous chapter.

Chapter 10 describes the OPNET testbed in which OPRA was implemented for experimental characterization.

Chapter 11 evaluates OPRA's ability to restore all single span failures in a metropolitan and a long haul network using the testbed presented in chapter 10.

Chapter 12 investigates the impact of reducing the processing delay associated with executing OPRA, compared to the value used in the previous chapter.

Chapter 13 reports the behaviour of OPRA when presented with a random series of individual link failures, as opposed to a "guillotine" (i.e. simultaneous) model of a cable cut.

Chapter 14 quantifies the benefits of the interference heuristic which enables OPRA to synthesize a near optimal multi-commodity max-flow restoration pathset in usually less than two seconds.

Chapter 15 investigates OPRA's ability to restore multiple span failures, and synthesize a restoration pathset which appoints to every affected demand pair an amount of spare capacity proportional to that network's pre-failure connectivity.

Chapter 16 concludes this thesis with a review of the main development, a summary of the research contributions made, and recommendations for further research.

# Chapter 2. Transport Networks

This chapter provides background on the networks whose capacity design will be optimized and into which the distributed dynamic path restoration algorithm will be deployed. The relevant network environment is that of the SONET/SDH transport networks. In general, transport networks must provide the physical connectivity, establishment, maintenance, and release of connections. The Synchronous Optical Network (SONET) defines the first layer, physical layer, of modern transport networks, and recently Asynchronous Transfer Mode (ATM) has emerged as a leading candidate to fulfil the services associated with the next two layers, the data link and network layers, of a transport network. The distributed dynamic path restoration algorithm proposed in this thesis dissertation can be deployed in either the SONET or ATM layer. Considering that SONET is a relatively mature technology compared to ATM, the current research investigates the performance of the distributed dynamic path restoration algorithm assuming a SONET technology platform.

The following sections review SONET and ATM, focusing on those aspects of each technology which facilitate mesh restoration. In this review the essential differences between SONET and ATM, and how those differences effect the functioning of a mesh restoration algorithm are highlighted.

## 2.1 SONET

High speed transport carrier signals are usually the end product of multiplexing many lower speed signals together. As a digital carrier signal traverses the transport network it is usually multiplexed and demultiplexed many times. Maintaining the visibility of low speed signals within the carrier signal is a goal achieved by SONET but not by the older Plesiochronous Digital Hierarchy (PDH), e.g. DS-1/2/3-based, transport networks. The following discussion introduces SONET networks by comparing them to PDH transport networks.

A Plesiochronous Digital Network (PDN) uses point-to-point transmission systems and a layered multiplexing scheme to provide the physical connectivity, establishment, maintenance, and release of connections. In a PDN framing occurs at each multiplexing step, and each point-to-point transmission system is clocked independently. While all the clocks in a PDN are free running, they nominally operate at one of the standardized rates set in the Plesiochronous Digital Hierarchy (PDH) shown in Table 2.1.

Table 2.1. Plesiochronous Digital Hierarchy (North America)

| Digital Signal Level | Data rate (Mbps) |
|---|---|
| DS0 | 0.064 |
| DS1 | 1.544 |
| DS2 | 6.312 |
| DS3 | 44.736 |
| DS4 | 274.176 |

Within each rate in the multiplexing hierarchy, the various transmission systems in a PDN operate at slightly different frequencies. In order to multiplex signals with slightly different bit rates it is necessary to adjust the various input signals to a common rate by adding or deleting bits. The PDH specifies specific positions in a signal where special bits, referred to as stuff bits, may be eliminated or added without corrupting the transmitted information. While adding and deleting stuff bits in an input signal according to the rules stipulated in the PDH does not corrupt the information being transmitted, it does render the tributary signal inaccessible after multiplexing. In the PDH it is impossible to discern the difference between a stuff bit and an information bit in the payload of any digital carrier signal above the DS0 level without demultiplexing the high speed signal into its constituent tributary signals.

In the Synchronous Optical Network (SONET) the payload remains visible after multiplexing, foregoing the need to completely demultiplex a high speed signal into its constituent parts in order to recognize the information being transmitted. Instead of using stuff opportunities like the PDN to achieve frequency justification among plesiochronous signals, SONET uses pointers to indicate the variable start position of each individual tributary signal within the payload of a synchronous high speed carrier signal. The synchronous portion of a SONET carrier signal used to transport a tributary signal is called the Synchronous Payload Envelope (SPE). SONET is able to retain payload visibility by using pointers to indicate where the information being transmitted begins within the base level synchronous transport signal. The base level synchronous transport signal in SONET is the STS-1.

Payload visibility allows for single-step multiplexing and demultiplexing which reduces the number of redundant terminations and interfaces at a node, and allows greater equipment integration in terms of physical size and application functionality. All low speed tributary signals in SONET can be multiplexed to a standardized high speed bit rate in a single step. Furthermore, while the tributary signals to a STS-1 may be synchronous or plesiochronous, SONET uses a universally distributed clocking signal to synchronize all transport signals. By slaving all clocks in a SONET network to one universal clocking signal, all STS-Ns are synchronized and the need for frequency justification between SONET transport signals is eliminated.

With the advent of Fibre Optic Transport Systems (FOTS) and bit rates much faster than those specified in the PDH, retaining payload visibility as well as defining signal formats for bit rates faster than DS-4 is necessary. SONET standardizes the use of pointers to maintain payload visibility as well as high speed carrier signal formats as shown in Table 2.2. Unlike the PDH, the bit rates of the higher levels signals in the SONET signal hierarchy are a multiple of the of the base level bit rate (i.e. STS-1).

Table 2.2. Levels of the SONET Signal Hierarchy

| Optical hierarchy level | Line rate (Mbps) | Digital hierarchy level | |
|---|---|---|---|
| | | ANSI ("SONET") | CCITT ("SDH") |
| OC-1 | 51.84 | STS-1 | - |
| OC-3 | 155.52 | STS-3 | STM-1 |
| OC-9 | 466.56 | STS-9 | - |
| OC-12 | 622.08 | STS-12 | STM-4 |
| OC-18 | 933.12 | STS-18 | - |

Table 2.2. Levels of the SONET Signal Hierarchy

| Optical hierarchy level | Line rate (Mbps) | Digital hierarchy level | |
|---|---|---|---|
| | | ANSI ("SONET") | CCITT ("SDH") |
| OC-24 | 1 244.16 | STS-24 | - |
| OC-36 | 1 866.24 | STS-36 | - |
| OC-48 | 2 488.32 | STS-48 | STM-16 |

### 2.1.1 Distributing Alarms in a SONET Network

The path, line, and section overhead added to the service payload of a SONET signal provide support for a host of modern operational control, configuration, and surveillance systems. The optimized distributed path restoration algorithm presented in this thesis is one of the applications which could be supported by SONET. The restoration algorithm uses the STS Path Alarm Indication Signal (AIS) supplied by SONET to activate the algorithm, and also uses SONET's line overhead to propagate its state indications to adjacent nodes.

The STS Path AIS is associated with the path overhead of a SONET signal, and is used to notify the path terminating equipment (PTE), i.e. the source and destination of a working path, of a failure. The location where a payload tributary signal is mapped into and removed from a SPE is the PTE. In contrast to a PTE, Line Terminating Equipment (LTE) is responsible for multiplexing STS-1 frames into STS-N frames and Section Terminating Equipment (STE) is responsible for preparing a STS-N signal for physical transport. A single piece of network equipment may perform the functions associated with a STE, LTE, and PTE. Referring to Figure 2.1, in the event of a unidirectional failure a Path AIS is generated as follows:

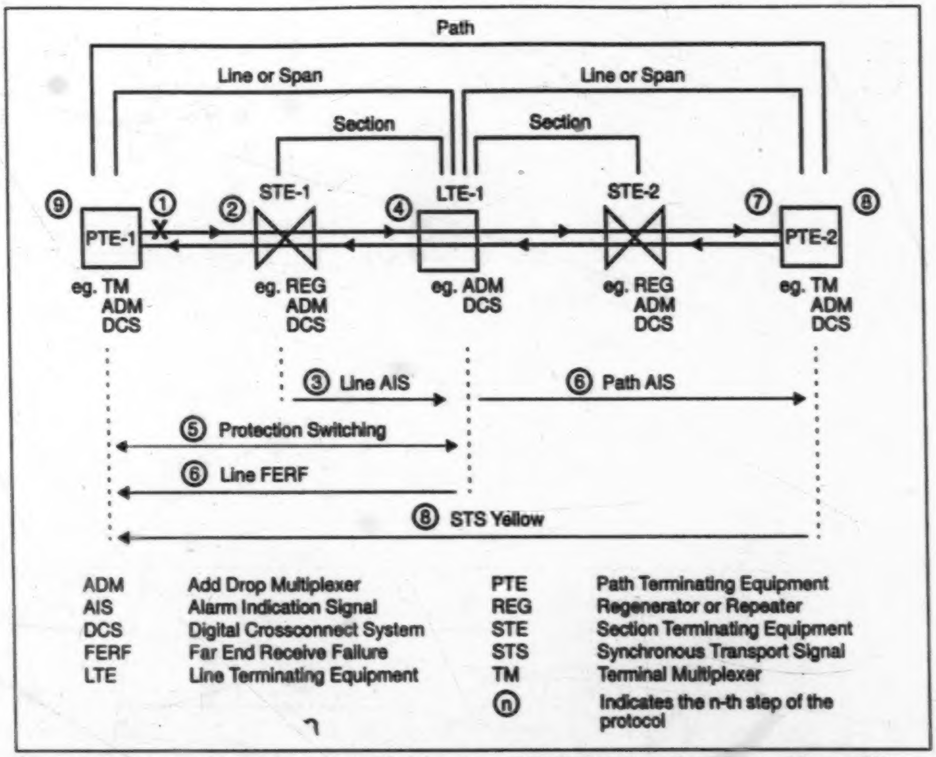

Path
Line or Span
Line or Span
Section
Section
⑨
①
②
STE-1
④
LTE-1
STE-2
⑦
⑧
PTE-1
PTE-2
eg. TM ADM DCS
eg. REG ADM DCS
eg. ADM DCS
eg. REG ADM DCS
eg. TM ADM DCS
③ Line AIS
⑥ Path AIS
⑤ Protection Switching
⑥ Line FERF
⑧ STS Yellow
ADM Add Drop Multiplexer
AIS Alarm Indication Signal
DCS Digital Crossconnect System
FERF Far End Receive Failure
LTE Line Terminating Equipment
PTE Path Terminating Equipment
REG Regenerator or Repeater
STE Section Terminating Equipment
STS Synchronous Transport Signal
TM Terminal Multiplexer
ⓝ Indicates the n-th step of the protocol

Figure 2.1. Path Alarm Distribution in a SONET Network [adapted from 69]

1. A unidirectional failure such as a partial cable cut occurs between PTE-1 and STE-1.

2. STE-1 enters a Loss of Signal (LOS) state 100 μsec after the onset of an all 0's pattern.

3. STE-1 generates a Line AIS to LTE-1 within 125 μsec after entering the LOS state.

4. LTE-1 enters a Line AIS state after receiving five consecutive SONET STS-1 frames with the Line AIS set.

5. LTE-1 attempts to restore the failure using APS or SHR systems in current practise. This is an aspect that may change with the present work.

6. LTE-1 alerts PTE-2 and the upstream LTE (PTE-1 in Figure 2.1) of the failure within 125 μsec after entering the Line AIS state by transmitting a Far End Receive Failure (FERF) signal and a Path AIS as shown.

7. PTE-2 enters the Path AIS state after receiving three consecutive SONET STS-1 frames with the Path AIS set.

8. PTE-2 transmits an STS Yellow alarm to PTE-1 upon entering the Path AIS state, and after waiting a minimum of 50 msec for any protection switching systems present to restore the failed working paths they were meant to protect, activates the distributed optimized path restoration algorithm for those demand pairs whose working paths have not been restored.

9. PTE-1 acknowledges the STS Path Yellow alarm after receiving ten consecutive SONET STS-1 frames with the STS Path Yellow alarm set. PTE-1 then waits a minimum of 50 msec to allow any protection switching systems in place to restore those transport signals they were meant to protect before activating the distributed optimized path restoration algorithm as described in step 8.

Given a bidirectional failure, alarms are distributed by the terminating equipment on either side of the failure following steps similar to those described above. However, unlike the unidirectional failure scenario, the optimized distributed path restoration algorithm is activated in both PTEs by Path Alarm Indication Signals rather than by Path AIS and STS Yellow alarm signals.

### 2.1.2 SONET Data Communications Channel (DCC)

In addition to distributing the alarms required to activate the optimized distributed path restoration algorithm, SONET facilitates communication between the nodes involved in a restoration event. While path level alarms as shown in Figure 2.1 will be required to activate OPRA, communication during a restoration event must occur at the line-level rather then the path-level because a solution to the restoration problem must specify precise crossconnections at a LTE between line-level signals. The Line Overhead (LOH) of a SONET transport signal includes a nine-byte, 576 kbps, Data Communications Channel (DCC) to facilitate message-based communications between nodes at the line-level.

When the DCC is used as the framework for communication between nodes, interactions take place by means of packetized messages explicitly addressed to another node. Packet contents are typically high-level constructions explicitly relating to commands or operations of a particular application. In traditional messaging if a node wishes to send a message it appends a destination address to the message. The message is then passed through the network from node to node. At each node, the entire message is received, processed, and then transmitted to the next node. Processing a message requires traversing a protocol stack to determine the command or operation that must be performed. A message is delayed at each node for the time required to receive all bits of the message, process the message, plus a queuing delay waiting for an opportunity to retransmit to the next node. The DCC can be used to carry messages of all types, and any node in the network can send messages addressed to any other node, not only adjacent nodes.

Message based communication using the DCC is not suited to real-time traffic. The delay through the network is relatively long and has relatively high variance. Considering that the objective of this research is to restore a failure in less than two seconds, message-based communications will be too slow, as found in [63]. Given the current performance characteristics of messaging-based communication on the DCC, coupled with the concerns of routing and message congestion at the time of a failure, using the DCC for signalling in DCS distributed restoration is not recommended [63].

### 2.1.3 State-based Signalling in a SONET Network

State-based signalling is the repeated application of static or quasi-static fields to individual links of a transport network such that a framework for interaction between nodes is established. State-based signalling is an attractive alternative to message-based communication because it is potentially much faster. The preceding work on the SHN presented in [25] showed that the persistent nature of state-based signalling and its space divided nature has benefits in simplicity and robustness when used to communicate during a restoration event because network state information is stored on the links of the network. These links function as memory locations, and the network fabric is the memory space within which an algorithm operates. There is no overhead required by the algorithm to communicate with an adjacent node or to determine the current logical state of a neighbour. The information is always available on the links terminated at the node.

As explained in [25], a network using state-based signalling as the framework for interaction does not require extensive or any explicit error correction. If one instance of an information field is errored, the receive hardware register holds its last valid value and accepts the next repetition of the field when correctly received with the checksum validated in hardware. There is no input message queuing; new information overwrites rather than follows the information in a field. With state-based signalling, every node interacts with its neighbours through highly parallel means, exploiting each link leaving its site for coupled memory-like interaction over fast, dedicated physical-layer channels.

Simple dedicated hardware in the DCS port and one or both of the unused bytes in the LOH of a SONET transport signal is all that is required to facilitate state-based signalling in a network. The optimized distributed path restoration algorithm presented in this thesis uses state based signalling to communicate during a restoration event as recommended in [63].

### 2.1.4 SONET DCSs

SONET specifies the transmission and multiplexing standards required to send information in a transport network. The hardware that terminates the digital signals and automatically cross-connects their constituent (tributary) signals is the Digital Cross-connect System (DCS). DCSs provide the flexible network control required to implement a mesh restoration mechanism.

A DCS historically provides a defined interface point at which standard impedance, power levels, digital waveshapes, and timing jitter are guaranteed so that all items of equipment that source, sink, or transport a digital carrier signal can be interconnected in any manner desired. The DCS, which is designed to integrate signal add-drop, cross-connect, multiplexing/demultiplexing, etc. into a single piece of network equipment, can eliminate back-to-back multiplexing and reduce the need for intermediate electrical distribution frames.

A variety of DCS machines have been developed for various applications and a standardized notation has arisen to describe a DCS. The notation DCS n/x denotes two attributes: n is the highest digital hierarchial transmission level at which the DCS machine can interface to transmission equipment and x is the lowest digital level at which the DCS can switch sub-multiplexed tributaries.

The functional diagram of a SONET DCS that switches digital signal levels slower than the digital hierarchical transmission level interfaced (i.e. $x<n$) is shown in Figure 2.2. When a high-speed signal is terminated at the DCS, it is first demultiplexed into a set of tributary channels. These tributary channels are then cross-connected to appropriate output ports for either service or facility grooming, or restoration. The tributary channels that are switched to the same output port are multiplexed together and converted to a high-speed, time division multiplexed, stream. This stream is then sent to the appropriate destination over a fibre facility. The cross-connect matrix, which performs channel switching, is managed by a controller that changes the switching matrix, as necessary.

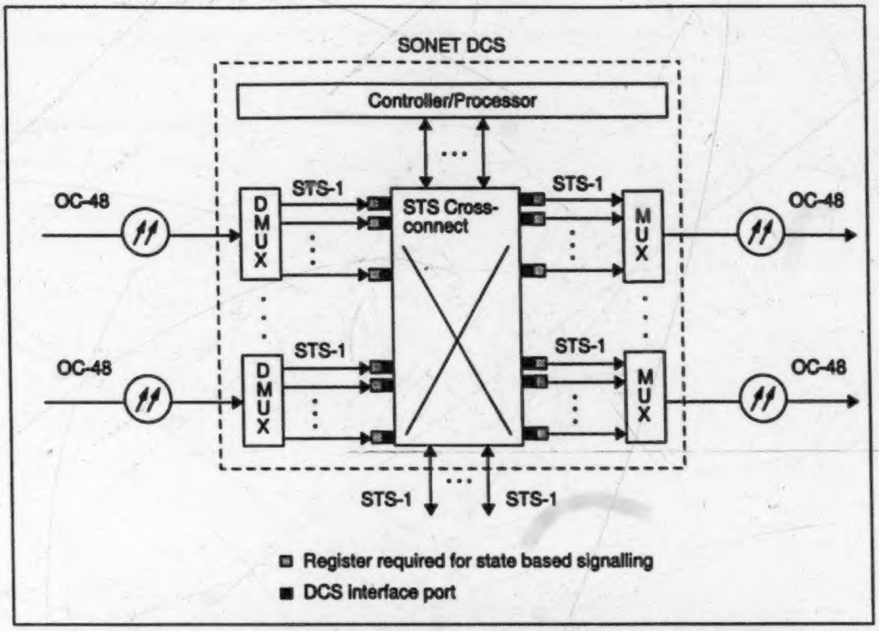

SONET DCS
Controller/Processor
OC-48
OC-48
D M U X
D M U X
STS-1
STS-1
STS Cross-connect
STS-1
STS-1
M U X
M U X
OC-48
OC-48
STS-1
STS-1
Register required for state based signalling
DCS interface port

Figure 2.2. SONET DCS functional diagram with enhancements to support state-based signalling [adapted from 25]

Figure 2.2 also shows the enhancements required if a DCS uses state-based signalling as the framework for interaction between nodes. As shown, each port requires a register capable of storing, receiving, and transmitting the information fields associated with state-based signalling. Each port register is provisioned to store and transmit the fields of one outgoing signal, and receive and store the fields of one incoming signal.

From the DCS processor point of view, the register is static memory which can be used to modify the fields used for state-based signalling on the outgoing signal of a port. The fields of a register modified by the DCS processor are continually repeated on the outgoing signal by special circuits in the port card.

In addition to altering the state of an outgoing signal at a port, the register shown in Figure 2.2 retains the status of an incoming signal. The state of a link is extracted from the line signal at a port by a serial-in parallel-out register, and stored in a second parallel-in parallel-out register. The serial-in parallel-out register accumulates individual bits from the incoming line signal and serially shifts them past circuits that recognize the beginning and end of an information field, as well as verifying the integrity of the field using either simple persistence checking or a checksum calculation. When a complete and verified information field is received in the first register, and it is different from the field in the second output register, the new value is latched into the output register and the processor informed of the change. The second register always holds the last valid state of a field for presentation to the DCS processor. A more complete description of the registers required to facilitate state-based signalling is given in [26]

The functional architecture of the DCS as shown in Figure 2.2 is currently under investigation. The present generation of DCS machines are not designed with fast restoration in mind. Historically, DCSs were seen to fulfil the role of provisioning and transport network rearrangement only. This is reflected in the current specification of 1 second cross-connect times [28]. During routine provisioning and rearrangement on DS3/STS-1 streams there is little reason to hurry and every reason to be cautious. However, in distributed restoration, connections are deduced locally within each node during an emergency, and specifications for normal crossconnection times should not apply. In order to reduce the restoration time, a prompt-connect specification is necessary during restoration. A prompt connection can be considerably faster than 1 second with only software changes primarily to waive spare port testing and connection verification, in which case cross-connection is technically feasible in 20 to 40 msec. [28]

Furthermore, present DCS architectures may use serial message processing and serial path cross-connection. By switching to a parallel CPU-based processing architecture and a parallel path cross-connection capability, a DCS's performance may be enhanced such that one STS-1 path is cross-connected in 10 msec [70].

It may be difficult to restore a network before a call in progress is dropped using present DCS system architectures if crosspoints cannot be operated quickly enough either due to large workloads during a restoration response or because of implementation choices. As a result, research that is being conducted in parallel with this work suggests that it can be useful to pre-operate restoration crosspoints between spare links before any failure has occurred, putting the network into a statistically optimal state of readiness [50]. When a failure occurs some of the preconfigured restoration path bundles can be used immediately. If more restoration paths are needed, they can be obtained by the optimized distributed path restoration algorithm presented in this thesis. The first advantage of preconfiguration is that the number of crossconnection operations may be greatly reduced or eliminated for a portion of the affected traffic. This will reduce restoration time. Second, after utilizing preconfigured restoration paths, the workload of a real-time restoration algorithm will be lower because it will be searching for fewer paths. Preconfiguration as explained in [50] is not related to the functionality or performance of a distributed restoration algorithm, and does not alter the way OPRA would be used.

The time required to cross-connect a STS-1 path depends on the architecture of a DCS and the number of preconfigured restoration path bundles available. The cross-connect time is independent of the mesh restoration algorithm used to restore a failure. Therefore, the restoration times recorded in all of the results presented in this thesis do not include the time required to cross-connect a STS-1 path. The restoration times presented are the times required to complete the last restoration plan for all the nodes in the network restoration pathset.

It is difficult to predict the time required to complete the last cross-connection for all the nodes in the network restoration pathset from the restoration times presented in this thesis because the restoration pathset is built up as restoration paths are found. Considering that restoration paths identify the cross-connections a node must implement to restore lost working paths, and are rarely all found at once, a DCS may begin completing cross-connections before a node's entire restoration plan is realized. For example, if restoration paths are found sequentially, and the time interval between each find is greater than the time it takes for a DCS to physically complete a single cross-connect, the time at which all of a node's cross-connections are completed after the start of a restoration event is equal to the time that the final cross-connection at a node is completed. Only when the time interval between finding restoration paths is greater than the time it takes for a DCS to physically complete a single cross-connect will the restoration times presented in this thesis include the time required to complete all of the cross-connections for all of the nodes in the network.

## 2.2 Relationship to ATM

SONET networks provide a high-capacity transport system with powerful network operations and maintenance capabilities. However, traditional synchronous time division techniques used in SONET networks, which divide bandwidth into a number of fixed capacity channels, require all data to be transported at one of the standardized transmission rates shown in Table 2.2. These characteristics make SONET most efficient at supporting Constant Bit Rate (CBR) services such as voice service, which map well into the available fixed-capacity channels. Thus, SONET alone may not be adequate to support B-ISDN because it does not provide sufficient flexibility to manage a variety of bandwidth requirements within the same network. B-ISDN will allow for the efficient provisioning of telecommunications services to business and residences by providing a common, high-speed interface that can integrate voice, data, graphics, and video information.

The technology platform proposed to carry B-ISDN is a packet-oriented transfer mode known as Asynchronous Transfer Mode (ATM). The packets used to organize and transport information in an ATM network are fixed-size entities known as cells. ATM is asynchronous in the sense that the recurrence of cells containing information from an individual user are not necessarily periodic. ATM technology combines the flexibility of traditional packet-switching technology, which is efficient for supporting services with various bandwidth requirements, with the determinism of TDM, which is suitable for supporting services requiring stringent delay requirements.

Both the SONET and ATM network architecture for switched services are made up of two switches at the ends of a path and a transport network which conveys signals through intermediate nodes such as DCSs. In both cases, all major call processing and management functions are performed at the two end nodes of a path, and the transport network is designed to deliver signals as quickly and as cost-effectively as possible with a minimum amount of processing.

As shown in Figure 2.3, the ATM switch, which is composed of a high-throughput cell switch and a high-performance processor, handles VP assignment, setup, processing, and capacity allocation, as well as VC routing. ATM DCSs, which are used for provisioning VPs in broadband transport networks as well as for VP routing, simplify network configuration and provide flexible routing and capacity allocation.

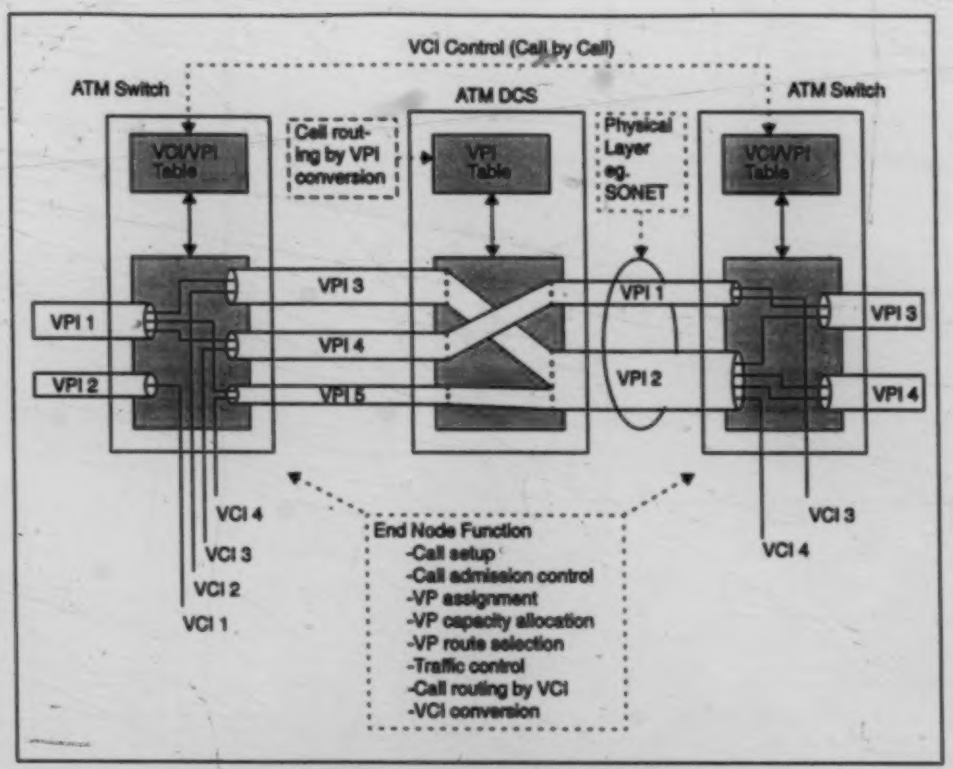

VCI Control (Call by Call)
ATM Switch
ATM DCS
ATM Switch
VCI/VPI Table
Cell routing by VPI conversion
VPI Table
Physical Layer eg. SONET
VCI/VPI Table
VPI 3
VPI 1
VPI 1
VPI 4
VPI 2
VPI 2
VPI 5
VPI 3
VPI 4
VCI 4
VCI 3
VCI 2
VCI 1
End Node Function
-Call setup
-Call admission control
-VP assignment
-VP capacity allocation
-VP route selection
-Traffic control
-Call routing by VCI
-VCI conversion
VCI 3
VCI 4

Figure 2.3. ATM VP-Based Network Architecture and Nodal Functions [adapted from 69]

Several applications of the VP concept exist. VPs provide a logical direct link between two nodes. This simplifies signal transport between two nodes. In addition, VPs can provide protection switching or alternate routing in case of a failure. Features of ATM VP technology which facilitate network restoration include non-hierarchical multiplexing (as shown in the ATM DCS in Figure 2.3), hidden paths, and Operation and Management (OAM) cells.

A hidden path is a VP that can be preassigned without reserving any capacity for it. Unlike a SONET DCS, which can only support path cross-connection with a single bit rate, the ATM DCS can cross-connect VPs with any bandwidth.

The OAM cell facilitates communication between nodes in an ATM network, analogous to the DCC in SONET. A restoration mechanism deployed in an ATM network can broadcast and acknowledge path rerouting signals using OAM cells. OAM cells can be inserted into the main ATM cell stream at any time as long as there is available capacity. OAM cells are suitable for message based signalling as described in section 2.1.2, but are unable to support state-based signalling as described in section 2.1.3. Table 2.3 highlights the differences between SONET and ATM mesh survivable networks.

Table 2.3. Comparing Mesh Survivable SONET and ATM Networks

| | SONET | ATM |
|---|---|---|
| Protection Level | STS-path | VP |
| Equipment | SONET DCS | ATM DCS |
| Path being cross-connected | digital path (physical) | virtual path (logical) |
| Path capacities that can be cross-connected* | a single STS rate defined in the SDH* | any* |
| Multiplexing hierarchy | SDH | none |
| Framework for interaction between nodes | message or state-based signalling | cells** |

*A SONET DCS can cross-connect only STS signals defined in the SDH which have the same rate, whereas any VP with a capacity ranging from zero to the line rate can be cross-connected by an ATM DCS.

** Whether ATM cells are suitable for signalling in DCS distributed restoration requires further research.

A restoration mechanism deployed in an ATM VP-based network may use concepts similar to those used by a restoration mechanism deployed in a SONET based network. Like its SONET counterpart, a mesh ATM VP-based restoration mechanism can be controlled in a centralized or distributed manner, classified as either a span or path restoration mechanism, and be implemented in a preplanned or dynamic fashion.

Several restoration mechanisms have been proposed for ATM VP-based transport networks using ATM DCSs [21, 42, 44]. Most of the ATM restoration algorithms proposed to date rely on a pre-plan of back-up VPs which are assigned zero bandwidth when idle. While the routing problem is addressed in these mechanisms, the span capacities required to support the sudden volume of cell traffic generated by active back-up VPs, without significantly increasing cell delay, buffering, and loss, has not. While no cells travel along a backup VP in [21,42, 44] during normal operation (no failures), ultimately the physical bandwidth to carry the restoration plus ordinary traffic must be present along the backup route if one is to rely on it in an emergency. Suitable bandwidth margins must be maintained on all links that may be required to bear a combination of working and backup VP routes so that if a failure occurs the bandwidth on those links can accommodate the sudden increase in traffic. None of the proposed ATM restoration mechanisms take into consideration the impact an active back-up VP with non-zero bandwidth will have on the Grade of Service (GOS) of an ATM network.

Furthermore, restoration traffic flows in an ATM network will be of a very different nature than existing traffic models: they will appear as sudden step-like increases in aggregate cell flow, already statistically multiplexed as a composite of individual sources. Given that restoration flows correspond to shifts in flows occurring within the body of the transport network itself, these demands are not subject to, or regulated by, the usual source-feedback and/or source throttling flow control mechanisms such as the leaky bucket. Therefore, ATM restoration might present a new and quite specialized traffic source "model" to be incorporated into the engineering of link utilization and buffer depths etc.

Even though ATM uses cells to transport information, a standardized physical layer bit stream, like SONET, will be used to transmit them. In an ATM network that uses SONET as the physical layer, the payload of the SONET SPE is used to transport ATM cells, and there is no difference between SONET and ATM payload restoration from a user's perspective. Furthermore, ATM networks are still in their infancy relative to SONET networks, and the debate about which layer (either the ATM or SONET layer) of a transport network should be used to restore a failure is currently unresolved. Given these circumstances, the description of OPRA presented in this thesis fits into a SONET level, rather than an ATM level, restoration strategy.

# Chapter 3. Restoration as a Routing Problem

A solution to the restoration problem, while providing affordable service continuity, requires specifying precise connections between links in adjacent spans. This chapter formally states the restoration problem using graph theoretical terms and explains the differences between restoration and the packet routing and call routing problems. This chapter begins with a review of the technical terms defined in graph theory required to formally state the restoration problem and concludes with a discussion of the complexity of the restoration problem. A similar discussion on this issue is found in [25]. This chapter revisits these issues due to their importance here and extends their consideration to the case of path restoration.

## 3.1 Describing Transport Networks using Graph Theoretical Terms

A transport network consists of a finite set of nodes, and a finite set of directed links. A directed graph consists of a finite set of vertices and a finite set of arcs. The graph theoretical entities of vertices and arcs are equivalent to the nodes and directed links of a transport network. A directed link, *x*, is described as an ordered pair (*u*,*v*) where *u* and *v* are the nodes directly connected by the directed link. Node *u* is where the directed link begins, and node *v* is where it terminates. A common graph theoretical equivalent is that *x* has its tail at *u* and its head at *v*.

An edge is a pair of directed links (arcs) between two nodes which have opposite directions. The common term edge in graph theory is analogous to the term link in this work, albeit there are many links on each span in general as this is the multigraph nature of the present problem. For a link *x*, *c(x)* is the length, distance, cost, or weight of *x*. A link is said to be positive or negative based on the sign of its weight. In the restoration problem, links have only positive weights.

A reflexive graph has a self-loop at every node; an anti-reflexive graph has no self-loops. In a symmetric graph, every (*u*,*v*) directed link has a return directed link (*v*, *u*). An anti-symmetric graph has no two nodes sharing such a pair of directed links. Transport networks are inherently anti-reflexive and symmetric. Because a single link is composed of a pair of directed links, it is helpful at various times to analyse transport networks as undirected networks of links between nodes or as symmetric directed networks with pairs of directed links between nodes as shown in Figure 3.1.

A graph is simple if it is anti-reflexive and symmetric and has at most one pair of links between any two nodes, each pair having the same endpoints but opposite orientation. The specialized form of graph needed for representation of the restoration problem is called a multigraph. A multigraph is an anti-reflexive, symmetric network in which any pair of nodes can be joined by any number of symmetric link pairs in parallel as shown in Figure 3.1.

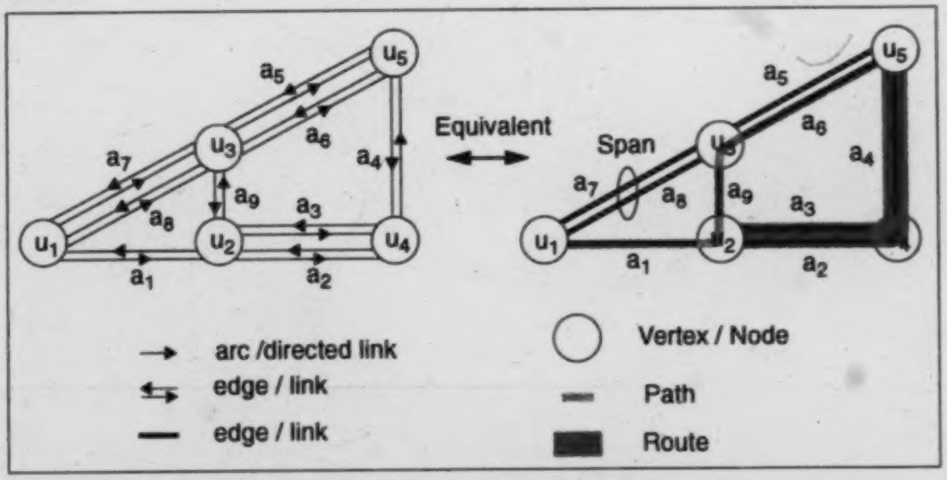


Figure 3.1. An anti-reflexive symmetric multigraph, i.e. a transport network

As shown in Figure 3.1, a path is an ordered set of concatenated links that may be represented by the set of links present in the path: $[P] = [a_1, a_9, a_6]$, or as a vector $P = (u_1, u_2, u_3, u_5)$ where $u_1$ is referred to as the source or origin and $u_5$ is called the target or destination. The length of path P is given by

$$C([P]) = \sum c(a_i) \ (\forall a_i \in [P]) \qquad 3.1.1$$

A path whose endpoints are distinct is said to be open whereas a path whose endpoints coincide is a closed path or a cycle. A simple path does not use any link more than once, and a loopfree path does not include any node more than once. Every loopfree path is by implication simple. In restoration each path found must be open and loopfree.

## 3.2 Relating Path Restoration to Call and Data Routing

Both span and path restoration differ from packet data and call routing in similar ways. Therefore a development similar to the one presented in [25] which addressed the span restoration problem is followed here, and a formal statement of the path restoration problem is developed starting with a comparison to packet data and call routing.

There are a number of algorithms for packet data and circuit switched networks that accomplish distributed routing, but use essentially centralized computational methods to derive the routing plan. These schemes either:

(a) perform routing calculations at a central site using a global view of the network and download the results to each node [60, 64], or

(b) include a scheme of mutual information exchange through which all nodes of the network first build a global view, then execute an instance of a centralized algorithm, and extract the part of the global routing plan that applies to their site [56, 60].

In either approach the final routing policy is put into effect in a distributed manner, but there is no truly distributed computation of the routing plans. A distributed implementation of a routing plan as described above is distinct from a distributed computation of a routing plan. The real-time demands of the restoration problem motivate the search for a scheme where computation and route deployment are both part of a single distributed process involving only isolated action by every node.

The algorithms that determine routings which optimize delay or blocking in packet and call routing can be quite sophisticated. Ultimately, however, the solution can be specified for any (origin, destination) pair (A,Z) in the form of one route in a simple network graph:

$$r_l = (A, u_2)\,(u_2, u_3)\,(u_3, u_4)\,\dots\,(u_{N-1}, Z) \qquad (3.2.1)$$

where $u_i$ is a node of the network, and $r_i$ specifies a loopfree route in a simple graph which is the sequence of nodes through which packets will be relayed. There is no constraint on the simultaneous use of any link in this route by different node pairs. All routings can re-use and access any link through delay queuing or blocking contention.

In dynamic call routing, the call attempt replaces the data packet, and trunk groups, accessed through blocking contention rather than delay queuing, replace the links of the packet routing problem. In this case $r_i$ specifies the sequence of trunk groups over which a new call attempt should be successively offered for call establishment. Otherwise, the situation is the same in terms of the information required to specify a routing solution. In this case, delaying an attempt is not usually an option. Individual calls must be allocated to specific links in the sequence of trunk groups that they occupy.

For path restoration a link is dedicated to at most one transport carrier signal. Path restoration of a span cut requires dedicated unique replacement paths between the source and destination of every failed working path. The target number of restoration paths is equal to the number of failed working links. Restoration attempts to allocate to each relation enough restoration paths to replace the lost working links. Each restoration path is dedicated to the logical replacement of one failed working path. The routing of replacement paths as a group must be consistent with finite capacity limits on each span. Unlike packets or calls, there is no equivalent form of blocking-based, contention-based, or delay-based shared access to links on a span. A carrier signal completely fills a restoration path in space, time, and frequency. The restoration path-set must therefore be:

(a) mutually link disjoint throughout, and

(b) collectively consistent with the capacity of each span of the network.

In contrast to packet and call routing where a sub-optimal routing state may mean some excess delay or blocking (graceful forms of degradation), the penalty for failure to satisfy (a) and (b) in restoration is abrupt and complete outage for carried services. Table 3.1 is a summary of the comparative aspects of call, packet, and restoration routing.

Table 3.1. Comparative aspects of packet, call, and restoration routing problems [adapted from 28]

| | Packet routing | Alternate call routing | Path restoration |
|---|---|---|---|
| Span structure | a single logical "pipe" | a single logical trunk group | multiple links<br>e.g. DS-3s or STS-1s |
| Span capacity allotment scheme. | queuing<br>i.e. time sharing | blocking<br>i.e. space sharing | restoration algorithm dedicates individual links in time and space to individual replacement paths |
| Grade of service (GOS) metric | path delay<br>(delay engineered) | call blocking<br>(traffic engineered) | network availability<br>(restorability engineered) |
| Type of network graph | simple | simple | multigraph |
| Application | data communication | voice telephony | physical bearer service for all voice and data services |
| Impact of extended failures (i.e. greater than 2 seconds) | increased delay | increased blocking of originating calls | loss of all calls in progress plus service outage |

## 3.3 Routing Formulation for Path Restoration

Path and span restoration both require identifying a set of link disjoint paths in a mesh restorable network, and the routing formulation for path restoration presented here parallels the development presented in [28] for span restoration, using similar notation. Most differences result from the fact that variables in a routing formulation for path restoration often are associated with a specific relation, requiring that a superscript, *AZ*, which identifies the demand pair to which a variable belongs, be added to the notation used in [28].

A solution to the path restoration problem for span (x, y) must include the following information for each (A,Z) demand to be restored:

$$K_{xy}^{AZ} = min\left(W_{xy}^{AZ}, T_{xy}^{AZ}\right) \qquad 3.3.1$$

$$R_{xy}^{AZ} = [r_1 \ldots r_k] \qquad 3.3.2$$

where:

$K_{xy}^{AZ}$ = number of restoration paths used by relation A-Z to restore links lost upon the failure of span x-y, sometimes shortened to $k_s^r$ where r is a tag for demand pair A-Z, and s is a tag for span *xy*,

$X_{xy}^{AZ}$ = number of working paths from relation A-Z which traverse the failed span x-y,

$T_{xy}^{AZ}$ = number of link disjoint restoration paths that are topologically feasible between node pair A-Z, combined with those restoration paths from every other demand pair to be restored, over the unused spare links in the network after the failure of span x-y,

$R_{xy}^{AZ}$ = set of $K_{xy}^{AZ}$ restoration paths, of various lengths, with each path $r_i$ defined as:

$r_i = [(A, u_{i,1}), s_{i,1}, l_{i,1,1}], [(u_{i,1}, u_{i,2}), s_{i,2}, l_{i,2,1}] \dots [(u_{i,h(i)-1}, Z), s_{i,h(i)}, l_{i,h(i),1}]$

where:

$u_{i,a}$ is the a<sup></sup>th node traversed by restoration path i

$s_{i,b}$ is the bth span traversed by restoration path i

$(A, u_{i,a})$ are the nodes terminating span $s_{i,b}$

$l_{i,b,c}$ is the cth link used on the bth span traversed by restoration path i

$h(i)$ is the logical path length of restoration path i.

$R_{xy}^{AZ}$ is called the restoration plan for node pair (A,Z) in the event that span x-y fails. $r_i$ details all the information required for a single restoration path, specifying exactly which links in adjacent spans must be cross-connected. The cross-connect information for each replacement path is specified at each node because restoration manipulates the individual links within a span.

Determining the restoration plan for each node to be restored is complicated by the fact that the following constraints must be simultaneously satisfied:

**Constraint 1: Link Disjointness**

A single link can only be used once in a single restoration path:

if $l_{i,b,c} \in r_i$ then $l_{i,b,c}$ can appear only once in $r_i$ and $l_{i,b,c} \notin r_j$ for all $i \neq j$.

**Constraint 2: Span Capacities**

The number of restoration paths traversing a span must not exceed the number of spare links on that span:

$$\sum_{r=1}^{D_{xy}} \sum_{i=1}^{K_{xy}} \delta_{xy,b}^{r,i} \le |s_b| \quad \text{for all spans } b \neq xy \text{ and all span failures } xy$$

where

$D_{xy}$ is the total number of demand pairs affected by span cut $xy$

$\delta_{xy,b}^{r,i}$ takes the value of 1 if the i<sup></sup>th restoration path for demand pair r after the failure of span $xy$ uses span b, and 0 otherwise

$|s_b|$ is the number of spare links on span b

**Constraint 3: Mapping Preservation**

In addition to finding the required number of paths between relations, each end-to-end relation which uses more than one restoration path must share a common scheme of ordering the replacement paths to avoid transposition in the traffic streams substituted over the restoration paths. For path restoration, mapping preservation can be achieved by identifying the destination node of a restoration path to the source node, and correctly pairing receive and transmit signals using the STS ID included in the section overhead of every SONET signal.

**Constraint 4: Minimum Capacity Utilization**

Minimize the spare capacity used to restore a span failure:

$$Min\left\{\sum_{r=1}^{D_{xy}} \sum_{l=1}^{K_{xy}} C(r_l)\right\} \quad \text{for all span failures } xy$$

where

$D_{xy}$ is the total number of demand pairs affected by span cut $xy$.

$C(r_l)$ is the cost of path $r_l$.

Constraints 1 through 3 are essential for the basic function of restoration: constraint 1 says that the paths created must be fully link-disjoint, constraint 2 says they must respect the finite span capacities present, and constraint 3 says they must share a common end-to-end identifying scheme for correct traffic substitution. While it is essential that constraints 1 to 3 are satisfied, for an operational mechanism constraint number 4 is an objective, which will have to be substantially satisfied for industry to accept the solution generated by the restoration algorithm.

The routing formulation for span restoration can be derived from equation 3.3.1 and 3.3.2, and constraints 1 through 4, by replacing the relations to be restored AZ with the end-nodes of the failed span *xy* (e.g. $k_{xy}^{AZ} = k_{xy}^{xy}$). Compared to span restoration, determining $T_{xy}^{AZ}$ and satisfying contraint number 4 for path restoration is much more difficult. In path restoration, the replacement paths from each demand to be restored must be simultaneously feasible and their number globally maximized, which requires making efficient use of a network's spare capacity. $T_{xy}^{AZ}$ cannot be determined in isolation for a single relation and locally maximized, rather $T_{xy}^{AZ}$ should be set so that the amount of lost capacity restored is maximized globally over all demands to be restored. Furthermore, the solution to the restoration problem is not simplified when $T_{xy}^{AZ}$ is greater than $X_{xy}^{AZ}$ because contraint number 4 requires that the combined set of all replacement paths from each demand to be restored use as little spare capacity as possible.

In summary, the path restoration problem can be stated in graph theoretical language as follows: find a link-disjoint restoration pathset for every demand pair to be restored which minimizes the total spare capacity used in a multigraph subject to span capacity constraints with mapping preservation.

## 3.4 Complexity Considerations

In an attempt to compare the relative difficulty of solving the span and path restoration problems, this section analyses the theoretical complexity of various routing formulations for restoration. The computational complexities presented here, do not apply to any distributed restoration mechanism, rather they are intrinsic single processor views of the complexity of the basic problems. The centralized single-processor complexity of the various routing formulations gives some insight into each problem, and a relative indication of the difficulty associated with implementing a distributed restoration algorithm to satisfy each formulation.

Table 3.2 shows a progression in computational complexity as routing formulations become increasingly sophisticated. This table is similar to the one presented in [28], except for the value associated with a path restoration mechanism using a minimum amount of spare capacity. The fifth routing problem formulation is slightly different because it represents implementing path restoration via successive shortest paths for all possible orderings of demands, as explained below, instead of all permutations of individual demand relations, as explained in [28]. While each formulation is exponential in complexity and pertains to a path restoration mechanism that uses a minimum amount of spare capacity, minor disparities in complexity result from finding the solution in slightly different ways. The following development of the computational complexity of a path restoration mechanism is similar to the one presented in [28] in the first four cases, and only digresses in the fifth case to be consistent with the explanation presented here.

The first computational complexity shown in Table 3.2 is the complexity of finding a single shortest replacement route in a simple graph. This is equivalent to the call or packet routing problem of a single shortest path. The problem of finding a single shortest replacement path between two nodes of a network is of $O(n^2)$ complexity when solved using Dijkstra's shortest path algorithm [17], and O(nlogn) complexity using a binary minheap [48]. The first routing formulation in Table 3.2 can only be satisfied by a span restoration algorithm which implements a single route between the nodes terminating a failed span. This is the simplest approach but requires the most spare capacity of any mesh restoration scheme because all of the spare capacity needed to restore a span has to be in place on some single route.

Table 3.2. Computational complexity of various routing formulations [adapted from 28]

| Routing Problem Formulation | Computational Complexity* |
|---|---|
| 1 Find a minimum cost alternate route in a simple graph i.e. a trunking network | $O(n^2)$ |
| 2 Span restoration via k-shortest link disjoint replacement paths | $O(k_s \cdot n^2) = O(n^2)$ |
| 3 Span restoration via minimum cost maximum flow | $O(n^3)$ |
| 4 Path restoration via successive shortest paths analogous to Capacity Scavenging | $O\left(k_p \cdot n^2 \cdot \frac{n \cdot (n-1)}{2}\right) = O(n^4)$ |
| 5 Path restoration using a minimum amount of spare capacity | $O\left(k_p \cdot n^2 \cdot \left[\frac{n \cdot (n-1)}{2} \cdot w\right]!\right) =$<br>$O\left((n^2)^{n^2+1}\right)$ |

* *n* is the number of nodes in a network

The second formulation in Table 3.2 is for span restoration via k-successively-shortest link-disjoint paths, found by repeated application of a single shortest path algorithm. There is no increase in the basic order of complexity for this step. The worst case increase is a multiplier ($k_s$) equal to the maximum number of working links per span.

The third formulation represents span restoration with minimum cost maximum flow routing. Algorithms for Max Flow are of $O(n^3)$ to identify the maximum feasible flow between two nodes [24]. The pathset that realizes this flow must be found as a separate problem.

Computational complexity rises another order of magnitude when non-optimal "ad-hoc" path restoration, as explained in section 1.4.4, is used to restore a failure. Unlike span restoration, reconfiguration in path restoration is not bounded by a region surrounding the failure, rather each network demand affected by the span cut is re-routed from its source to its final destination. The complexity of the fourth formulation in Table 3.2 assumes that one span cut may affect $k_p$ working paths from every relation in the network. Theoretically $\frac{n \cdot (n-1)}{2}$ relations or demand pairs are possible in a network of n nodes. Since computing the k successively-shortest link-disjoint loop free paths between two nodes of a network has O($n^2$) complexity, the problem of finding k successively-shortest link-disjoint loop free paths between all node pairs has complexity $O\left(k_p \cdot n^2 \cdot \frac{n \cdot (n-1)}{2}\right)$. However, this is a naive form of path restoration because the total number of restoration paths feasible in a network depends on the order in which the pathsets are found. A distributed path restoration algorithm which satisfies this routing formulation would likely be sub-optimal in path number and spare capacity usage, and not reliably yield 100% restoration.

For path restoration with assured maximization of the total restoration capacity, the problem becomes a case of multi-commodity maximum-flow, and is exponential in complexity. All possible allocations of restoration paths need to be identified in order to find the optimal[1] restoration pathset. The fifth routing formulation presented in Table 3.2 is for a path restoration mechanism which instantiates path restoration via successive shortest paths for all possible orderings of demands to find a *near* optimal restoration pathset. If the average demand per node pair is $w$, the number of ways all demands can be restored is equal to the number of ways all the demand can be sequenced. If we have $\frac{n \cdot (n-1)}{2} \cdot w$ demands, they can be sequenced in $\left[\frac{n \cdot (n-1)}{2} \cdot w\right]!$ ways. The complexity of computing a near optimal restoration pathset by initiating path restoration via successive shortest paths for all orderings of the demands severed by a span cut is therefore $O\left(k_p \cdot n^2 \cdot \left[\frac{n \cdot (n-1)}{2} \cdot w\right]!\right)$.

The upper limits on the complexity of the various routing formulations presented in Table 3.2, derived above using restoration path counting arguments, indicate that only a distributed restoration algorithm like OPRA is likely to restore a failure within the two second call-dropping threshold. OPRA presented in Part II of this thesis.

---

1. Optimal restoration paths maximize the restorability of a network, Rn as defined in chapter 5, while using as little spare capacity as possible as explained in chapter 3.

# Chapter 4. Capacity Placement in Mesh[1] Restorable Networks

Solving the restoration problem not only requires restoring a failure but economically determining the capacity placement required to facilitate restoration. The total capacity required by a transport network to satisfy demand and protect it from failures contributes significantly to its cost. Methods have been developed to place spare capacity for span restoration in a network with a given set of working span sizes. [11, 36, 57, 66, 67] Path restorable networks can, however, be even more capacity efficient because the restoration problem is solved by end-to-end re-routing. This chapter describes a method for capacity optimization of path restorable networks and quantifies the capacity benefits of path restoration over span restoration. The further benefits of jointly optimizing working and spare capacity placement in path restorable networks are also quantified by extending the method. This chapter in conjunction with the optimized distributed path restoration algorithm solves the restoration problem and satisfies the goals established in section 1.4.4. This chapter begins with the development of a theoretical lower bound on redundancy for path restoration, and ends with the economical design of a network's capacity. Aside from the bounding considerations that follow, this chapter corresponds to the recently published work on this topic by the candidate [41]. The theory and results of this chapter not only support the further work on OPRA but constitute one of the main contributions of the thesis.

## 4.1 Lower Bounds on Spare Capacity Requirements in Span and Path Restorable Networks.

Redundancy is defined as the ratio of total spare to working capacity. A theoretical lower bound on the redundancy of 100% path restorable networks is useful to quantify the difference in capacity efficiency between path and span restoration, and determine whether path restoration is more or less capacity efficient than span restoration.

---

[1] These networks are called mesh restorable not to imply that the network is a full mesh, but to reflect the ability of the rerouting mechanism to exploit a mesh-like topology through highly diverse and efficient rerouting of failed signal units.

In the following analysis let:

S = number of spans in a network,
n = number of nodes in a network,
d = $2 \cdot S/n$ = average topological degree of nodes in the network (i.e in terms of spans not links),
$w_i$ = working capacity per span
w = average demand per relation severed by a span cut

$$= \frac{\sum_{i=1}^{S} w_i}{\sum_{i=1}^{S} (\text{number of relations routed over span i})}$$

W = average working capacity per span

$$= \frac{1}{S} \cdot \sum_{i=1}^{S} w_i$$

If we assume, for lower bounding considerations only:

a) network restoration is limited by the number of spares incident with the end-nodes of a failure,
b) each span carries the same capacity W,
c) each relation has the same demand w affected by a span cut,

then immediate egress for rerouting of failed capacity for the end-nodes of a failed span in a span restorable network is possible if the total spare capacity on the (d-1) unaffected spans terminated at these nodes is enough to accommodate the lost capacity W.

Given these assumptions, if span restoration is used to restore a failure as shown in Figure 4.2, each span must have W/(d-1) spares, making the minimum total spare capacity in the network:

$$C_S = \frac{W}{d-1} \cdot S$$

and the total working capacity in the network is:

$$C_W = W \cdot S$$

Therefore, the lower limit on the redundancy of a span-restorable network is

$$R_{span} = \frac{C_S}{C_w} = \frac{W}{d-1} \cdot \frac{S}{W \cdot S} = \frac{1}{d-1} \tag{4.1.1}$$

This is a well known result previously reported in [18] and [66]. A similar derivation for the case of path restoration using the same assumptions is presented next.

Consider using path restoration to restore a span cut and assume that the surviving portions of a failed path are not initially released. Unlike span restoration, most of the "failure nodes" in path restoration will still have access to all d spans during a failure because they are the end-nodes of the demands, distant from the failure location. However, the local egress capacity for rerouting a particular demand pair is still the sum of the spare capacity on the (d-1) unaffected spans, unless the surviving portions of a cut path are released. To clarify this point consider a node which uses a span on which its working traffic was disrupted to restore lost capacity. Then some other node in the network will be prevented from using two spans to restore its lost capacity. For example, as shown in Figure 4.1, if node A which is distant from a span cut uses the incident span on which it lost working capacity, named span y, for restoration, node Z which is adjacent to node A will not be able to access span y to restore its lost capacity even though node Z didn't lose any working capacity on span y. Therefore, even in path restoration, on average (d-1) spans must carry the lost capacity at each node, but now the end-nodes are many, and on average are concerned only with the portion of W that pertains to their individual demand pairs, w.

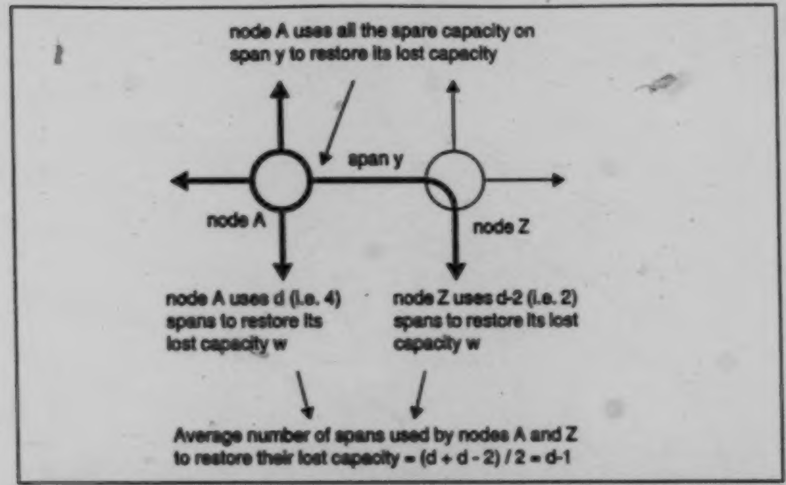

node A uses all the spare capacity on span y to restore its lost capacity
span y
node A
node Z
node A uses d (i.e. 4) spans to restore its lost capacity w
node Z uses d-2 (i.e. 2) spans to restore its lost capacity w
Average number of spans used by nodes A and Z to restore their lost capacity = (d + d - 2) / 2 = d-1

Figure 4.1. Average number of spans used per node to restore a failure

In a path restorable network it is advantageous to release the surviving portions of a cut working path and make those links available to the restoration process. This option is called stub release. Stub release is an option in a path restorable network but does not arise in a span restorable network because span restoration only replaces the cut portion of a connection as shown in Figure 1.4 in chapter 1, where as path restoration is equivalent to reprovisioning a completely new working path. In Figure 1.4 for example, the span cut shown leaves six stubs, two pairs of red stubs between A-S and C-T, and one black stub between D-S and B-T, which can be optionally released at the time of the failure and added to the pool of spares available for restoration.

If stub release is used, egress for rerouting of a particular demand pair would require that on average the sum of the spare capacity on d spans be greater than or equal to the lost capacity. In the following analysis, if the path restoration algorithm uses stub release, d-1 should be replaced with d.

If a span cut affects w units of demand per node, path restoration requires in this limiting case that each span have w/(d-1) spares to restore the failure. The total spare capacity in the network is then:

$$C_S = \frac{w}{d-1} \cdot S$$

while the total working capacity in the network remains:

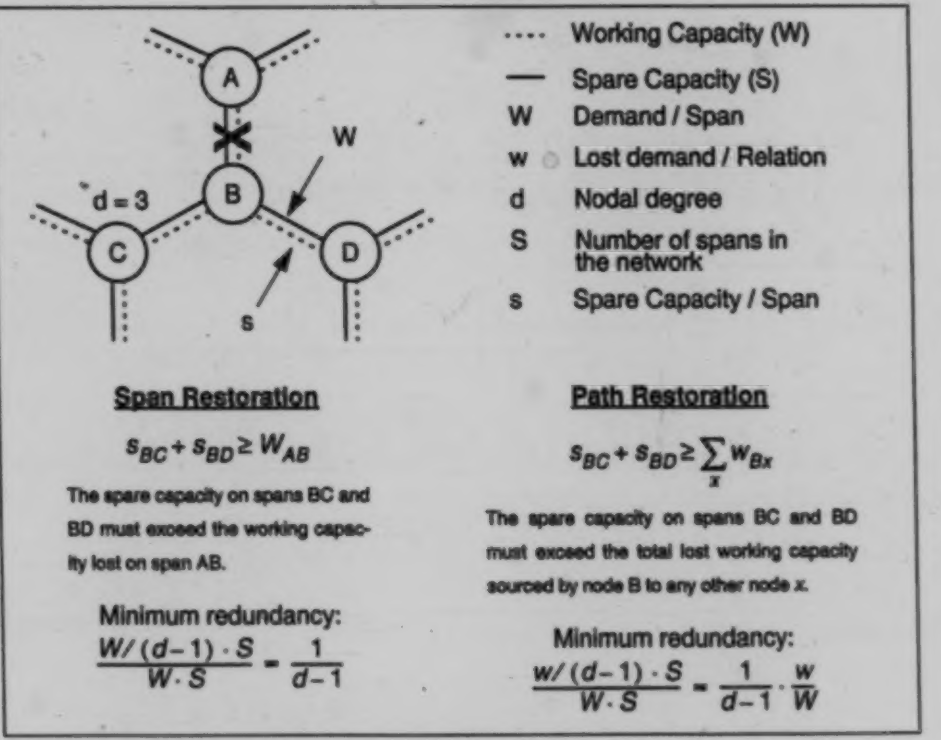
$$C_W = W \cdot S$$

Therefore, a lower limit on the redundancy of a path restorable network is:

$$R_{path} = \frac{C_S}{C_w} = \frac{w}{d-1} \cdot \frac{S}{W \cdot S} = \frac{1}{d-1} \cdot \frac{w}{W} \qquad 4.1.2$$

In other words the redundancy of a path restorable network benefits directly in proportion to the extent to which demands for each node pair are kept to a minimum on each span of the network. This can imply dispersion of demand routing, a concept returned to later.

Equations 4.1.1 and 4.1.2 are illustrated graphically in Figure 4.2. Note that equation (4.1.1) can be derived from equation (4.1.2). In span restoration only the end-nodes of a span cut are directly involved in restoring a failure in which case w = W and equations (4.1.1) and (4.1.2) become equivalent, as they should.



···· Working Capacity (W)
— Spare Capacity (S)
W Demand / Span
w Lost demand / Relation
d Nodal degree
S Number of spans in the network
s Spare Capacity / Span

**Span Restoration**

$$s_{BC} + s_{BD} \geq W_{AB}$$

The spare capacity on spans BC and BD must exceed the working capacity lost on span AB.

Minimum redundancy:

$$\frac{W/(d-1) \cdot S}{W \cdot S} = \frac{1}{d-1}$$

**Path Restoration**

$$s_{BC} + s_{BD} \geq \sum_{x} w_{Bx}$$

The spare capacity on spans BC and BD must exceed the total lost working capacity sourced by node B to any other node x.

Minimum redundancy:

$$\frac{w/(d-1) \cdot S}{W \cdot S} = \frac{1}{d-1} \cdot \frac{w}{W}$$

Figure 4.2. Lower bounds on redundancy in mesh restorable networks

A single span cut will sever at most W units of working capacity, so that $w \leq W$. Equation (4.1.1) is therefore an upper limit of equation (4.1.2), which implies the redundancy of a path restorable network should always be less than or equal to the redundancy of a span restorable network.

From equation (4.1.2) it is apparent that as w decreases, the lower bound on redundancy for path restoration decreases. This suggests that the average demand lost per relation can be minimized if working capacity is spread over many diverse routes. In this case instead of severing all of the demand from a few relations, a small amount of demand is lost by many relations.

Both w and W will change for various demand routing schemes and network topologies. More indirect routing schemes will increase the value of W and reduce w, while shortest path routing will minimize the value of W. Indirect routing schemes spread demands over many diverse paths in a network, decreasing w and increasing the number of nodes involved in restoring a span cut. Shortest path routing tends to maximize the value of w.

In addition, increasing the connectivity of a network can help decrease the value of W by concentrating less capacity on each span. The number of diverse paths over which demands can be spread will also increase, tending to decrease w.

In general it appears that the number of spares in a path restorable network is minimized in a fully connected mesh which employs an indirect demand routing scheme. Though decreasing w and increasing W would minimize the redundancy of a path restorable network, such a network may not be optimal[1] in terms of the total number of working and spare links required. The aim when designing a 100% path restorable network is not only to minimize the spare capacity, but to minimize the sum of the working and spare links required. Towards the end of this chapter the results from the path restorable network designs in the case of jointly optimized working and spare planning are inspected to see the extent to which this "dispersion" hypothesis may be manifested.

---

[1] The term optimal in this chapter implies a network which requires the minimum amount of working and spare capacity to restore all individual span cuts.

## 4.2 Integer Program Formulation

In this research Integer Programming (IP) is applied to the problem of optimal capacity planning in a mesh restorable network. A flexible IP formulation is presented here which can be used to optimize the placement of either the spare or working or combined capacity of either a span or path restorable network. The approach uses flow constraints based on a set of eligible predefined routes over which pathsets may be implemented. The solution to the IP tableau specifies the optimal capacity placement per span as well as the actual paths used to restore each possible span failure. The minimum capacity per span required to restore all individual span failures is of primary interest, and the restoration pathset information is of secondary importance. However, the restoration pathset information would be of primary importance if used to formulate a *centralized* restoration mechanism.

### 4.2.1 Integer Program Formulation for Spare Capacity Placement in Networks with Pre-defined Demand Routing

The following IP formulation optimizes the spare capacity placement of a path restorable network given a fixed working capacity design. The following notation, illustrated in Figure 4.3, will be used.

| | |
|---|---|
| Cj | Cost of a link (working or spare) assigned to span j. |
| S | Number of spans in the network. |
| $L_i^r$ | The restoration level required for demand pair r upon the failure of span i.<br>$0 \leq L_i^r \leq 1$ (for 100% network restorability $L_i^r = 1$ for all r and all i). |
| D | Total number of non-zero demand pairs in the demand matrix. |
| $D_i$ | Total number of demand pairs affected by span cut i |
| $d^r$ | Number of demand units between end-node pair r. |
| $X_i^r$ | Number of demand units lost by demand pair r upon the failure of span i. |
| $P_i^r$ | Total number of eligible restoration routes for demand pair r upon the failure of span i. |

| | |
|---|---|
| $f_i^{r,p}$ | The restoration flow through the p<sup>th</sup> restoration route for demand pair r upon the failure of span i. |
| $Q^r$ | Total number of working routes available to satisfy the demand between node pair r. |
| $g^{r,q}$ | The working capacity required on the q<sup>th</sup> working route to satisfy the demand between node pair r. |
| $\delta_{i,j}^{r,p}$ | Takes the value of 1 if the p<sup>th</sup> restoration route for demand pair r after the failure of span i uses span j, and 0 otherwise. |
| $\zeta_j^{r,q}$ | Takes the value of 1 if the q<sup>th</sup> working route for demand pair r uses span j. |
| $w_j$ | Number of working links (i.e. transport capacity units) on span j. |
| $s_j$ | Number of spare links on span j. |

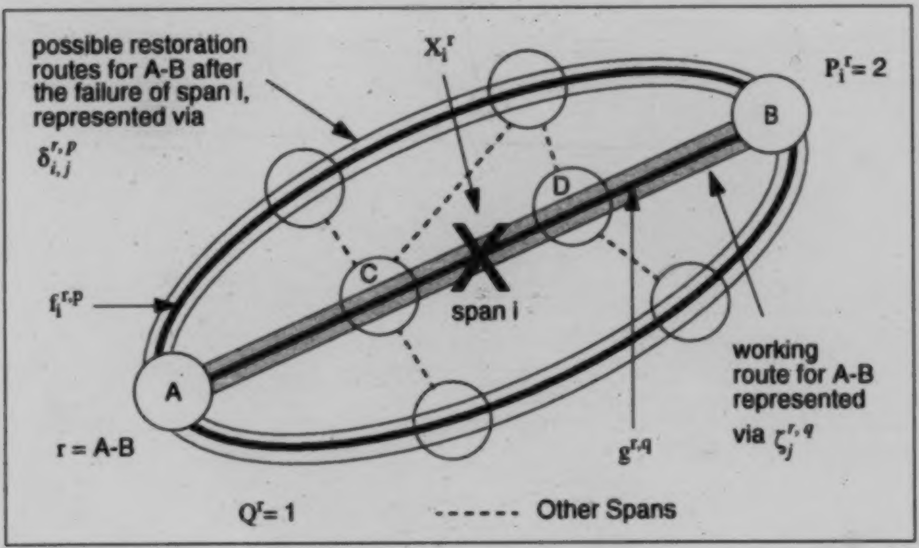

possible restoration routes for A-B after the failure of span i, represented via $\delta_{i,j}^{r,p}$
$X_i^r$
$P_i^r = 2$
B
D
C
span i
$f_i^{r,p}$
A
working route for A-B represented via $\zeta_j^{r,q}$
r = A-B
$g^{r,q}$
$Q^r = 1$
------ Other Spans

Figure 4.3. Integer Program Notation

The objective function is:

$$Min\left\{\sum_{j=1}^{S} C_j(s_j)\right\}$$

The constraints to be satisfied are:

1) Restoration flow meets target restoration levels for each demand pair r:

$$\sum_{p=1}^{P_i^r} f_i^{r,p} \geq \left\lceil X_i^r \cdot L_i^r \right\rceil$$

$\forall r = 1, 2, \ldots, D.$

$\forall i = 1, 2, \ldots, S.$

2) Span j's spare capacity is sufficient to meet the simultaneous demands of all restoration routes that use it to restore a single span failure:

$$(s_j) - \left(\sum_{r=1}^{D_i}\sum_{p=1}^{P_i^r} \delta_{i,j}^{r,p} \cdot f_i^{r,p}\right) \geq 0$$

$\forall (i,j) = 1, 2, \ldots, S.\ i \neq j$

3) The total demand lost by relation r after the failure of span i is the sum of the flows over relation's r working routes traversing span i:

$$\sum_{q=1}^{Q^r} \zeta_i^{r,q} \cdot g^{r,q} = X_i^r$$

$\forall r = 1, 2, \ldots, D.$

$\forall i = 1, 2, \ldots, S.$

4) The flows on restoration paths, $f_i^{r,p}$, and working paths, $g^{r,q}$, are non-negative integers.

5) Spare capacities, $s_j$, and working capacities, $w_j$, are non-negative integers.

As formulated, this IP can be adapted to optimize spare capacity placement for either a span or path restorable network. If a *span* restorable design is desired, the set of all node pairs affected by a failure is restricted to just the single pair of nodes terminating the severed span, i.e. $D_l = 1$, and $X_l^r = w_l$.

To represent stub release in the IP, constraint number 2 is augmented as follows:

2a) Span j's spare dimensioning is sufficient to meet the simultaneous demands of all restoration routes that use it to restore a single span failure (first double sum) after releasing the surviving portions of cut paths (second double sum).

$$(s_j) - \left(\sum_{r=1}^{D_l}\sum_{p=1}^{P_l^r} \delta_{i,j}^{r,p}\cdot f_i^{r,p}\right) + \left(\sum_{r=1}^{D_l}\sum_{q=1}^{Q^r} \zeta_j^{r,q}\cdot \zeta_i^{r,q}\cdot g^{r,q}\right) \ge 0$$

$\forall (i,j) = 1, 2, \dots, S.\ i \neq j$

Eligible working and restoration routes are specified in the IP constraint set using $\zeta_j^{r,q}$, and $\delta_{i,j}^{r,p}$ respectively. These routes are not themselves predefined restoration plans or predetermined working or restoration routes. They define the set of routes that could be used by the IP in order to optimize the capacity placement in accordance with the objective function. There will in general be many more eligible routes than actually used.

All distinct routes between demand pair end-nodes (or span end-nodes depending on the case) must be represented in the constraint system to find the minimum spare capacity placement. However, because the number of distinct routes in a network of $S$ spans is $O\left(2^{S}\right)$, the number of distinct routes entered as constraints typically has to be restricted in practice. The size of the route sets used to implement the above IP were restricted by limiting the length of eligible routes in a manner similar to the "hop-limited" approach described in [36]. Eligible routes could only be longer than the length of the shortest route by a limited number of hops and a given geographical distance limit. However, the use of hop and distance limits can prevent some long routes, which may be needed in some networks, from entering the route set. Consequently the hop and distance limited set of distinct routes was supplemented with the k-successively shortest disjoint routes between all pairs of nodes. This set of supplemental k-shortest disjoint routes between a node pair is relatively small, obviating the need for any distance limiting factors.

### 4.2.2 Integer Program Formulation for Combined Optimized Spare and Working Capacity Placement

The constraint system presented in section 4.2.1 is adequate for span and path restorable designs without jointly considering the routing of working demands before a failure. The IP can be extended to simultaneously optimize the working path routing *and* spare capacity placement of a path restorable network. An IP formulation which minimizes the sum of working and spare capacity must not only determine the spare capacity per span and the routing of all restoration paths, but also the working capacity per span and the routing of all working paths. By adding the following two constraints (6 and 7) to the IP formulation presented previously, the solution will include the values of $w_j$, and $g^{r,q}$ which will now minimize the *total* capacity-cost required in a path restorable network.

The objective function now becomes:

$$Min\left\{\sum_{j=1}^{S} C_j(s_j+w_j)\right\}$$

Subject to constraints 1 through 5 defined previously, and:

6) The total capacity on the working routes allocated to node pair r can carry all the demand of relation r:

$$\sum_{q=1}^{Q^r} g^{r,q} = d^r \qquad \forall r = 1, 2, \ldots, D.$$

7) Span j's working capacity is sufficient to meet the pre-failure demands of all relations which cross it:

$$(w_j)-\left(\sum_{r=1}^{D}\sum_{q=1}^{Q^r}\zeta_j^{r,q}\cdot g^{r,q}\right) = 0 \qquad \forall j = 1, 2, \ldots, S.$$

As before, the joint formulation can also be adapted to optimize the capacity placement in a *span* restorable network by designating the source and destination of all working paths cut by a span failure as the immediate end-nodes of the severed span (i.e. $D_i = 1$, and $X_i^r = w_i$), and eliminating stub release (i.e. using constraint 2, not 2a).

Furthermore, the IP formulation can be extended for the general case of multiple span and node failures by using the subscript i to identify the failure scenario rather than the failed span. In this case none of the constraints would be changed. However, the eligible restoration route set for different failure scenarios may be different because only those restoration routes which are topologically feasible in the failed network are included in the eligible route set for that failure scenario. For example, span j from restoration route p and relation r for failure scenario i, identified by $\delta_{i,j}^{r,p}$ in the eligible restoration route set, must not be cut by failure scenario i.

## 4.3 Networks Investigated

Five networks and demand matrices previously studied for span restoration were used to test the IP formulations presented in section 4.2. The characteristics of each network are detailed in Table 4.1 and the topology of each network is shown in Figures 4.4 to 4.8. The demand files associated with each network are presented in the technical report which accompanies this thesis [38]. Network one is a test network which has a uniform point-to-point demand matrix with two demand units between all node pairs. This is "SmallNet" as used in [25]. Network two is a metropolitan area model which was published with a demand matrix, used here, based on industry data [63]. Network three is another metropolitan area model based on a Canadian city. Networks four and five and their point-to-point demands are representative of long haul networks.

Table 4.1. Test Network Characteristics

| Network | No. of nodes | No. of spans | Avg. network degree | No. of pt-to-pt demands | Total amount of demand |
|---|---|---|---|---|---|
| 1 | 10 | 22 | 4.40 | 45 | 90 |
| 2 | 15 | 28 | 3.73 | 67 | 824 |
| 3 | 20 | 31 | 3.10 | 153 | 2 152 |
| 4 | 53 | 79 | 2.98 | 347 | 858 |
| 5 | 30 | 59 | 3.93 | 263 | 8 312 |

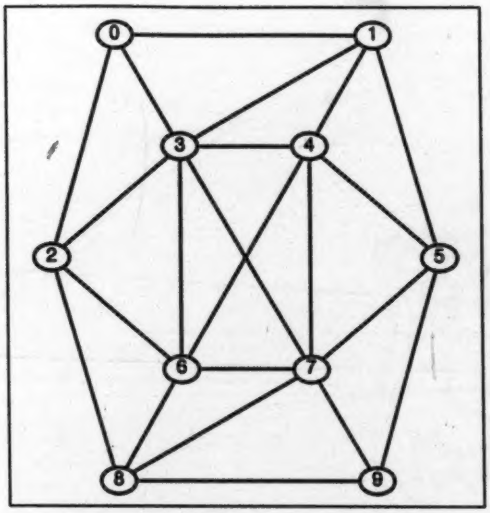

0
1
3
4
2
5
6
7
8
9

Figure 4.4. Topology of Network 1

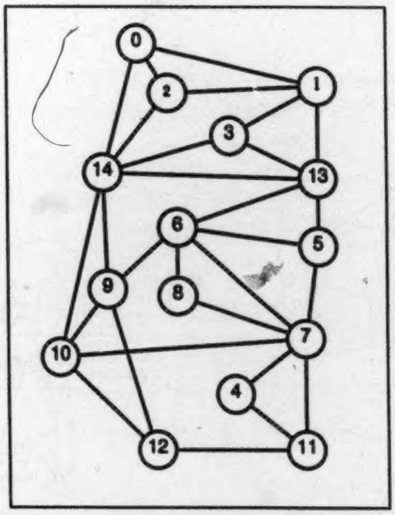

0
2
1
3
14
13
6
5
9
8
7
10
4
12
11

Figure 4.5. Topology of Network 2

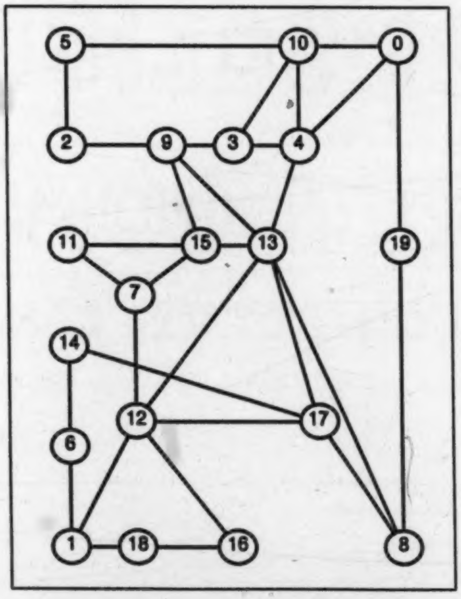

5
10
0
2
9
3
4
11
15
13
19
7
14
12
17
6
1
18
16
8

Figure 4.6. Topology of Network 3

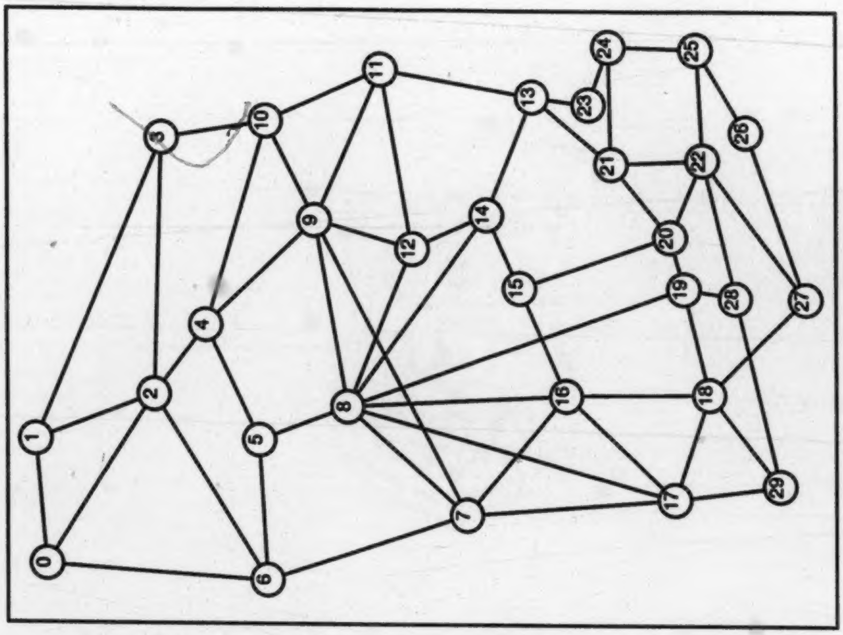

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

Figure 4.8. Topology of Network 5

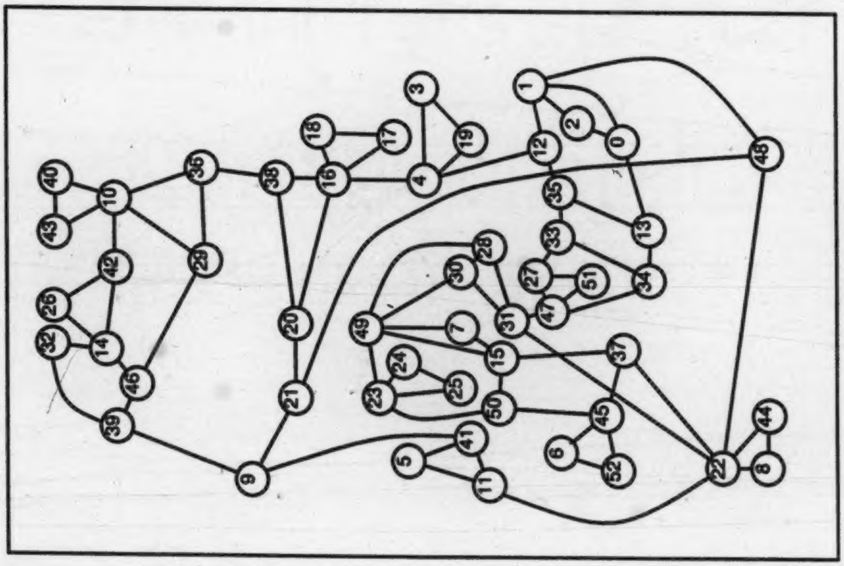

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52

Figure 4.7. Topology of Network 4

## 4.4 Capacity Placement Test Results

The capacity placement for each network detailed in section 4.3 was optimized to restore all single span failures. The first three cases optimized the placement of spare capacity in span and path restorable networks given a working capacity design. The working capacity design in cases 1, 2, and 3 split the routing of demand between a node pair as evenly as possible over the node pair's equally logically shortest disjoint routes. For example if

a) 7 units of demand need to be routed between nodes A and B,
b) the length of the shortest logical route between nodes A and B is 6 hops, and
c) five disjoint routes 6 hops in length, identified as routes 1 -5, are topologically feasible between node A and B,

two units of demand would be routed over route 1, two units of demand would be routed over route 2, one unit of demand would be routed over route 3, one unit of demand would be routed over route 4, and one unit of demand would be routed over route 5. The latter three cases, i.e. cases 4, 5, and 6, are repeats of the first three but with *jointly optimized* placement of spare and working capacity. The six cases are summarized below:

Case 1: Optimize the placement of *spare capacity* in a *span restorable network.*
Case 2: Optimize the placement of *spare capacity* in a *path restorable network without stub release.*
Case 3: Optimize the placement of *spare capacity* in a *path restorable network* with *stub release.*
Case 4: Optimize the placement of *working and spare capacity* in a *span restorable network.*

Case 5: Optimize the placement of *working and spare capacity* in a *path restorable network without stub release*.

Case 6: Optimize the placement of *working and spare capacity* in a *path restorable network* with *stub release*.

Figures 4.9 through 4.13 summarize the six case designs completed for each test network in terms of total network capacities. A sample design is presented at the end of this section. Details of each IP run, including execution times and the size of the route sets used by each tableau are presented in Tables 4.2 through 4.7. Details of each network design, including the number of spare and working links per span, are presented in the technical report which accompanies this thesis [38]. The correctness, of each design was verified using independent tools which tested that the working capacity satisfied all entries in the demand matrix, and that all failed working paths were 100% restorable using either span or path restoration, as appropriate to the case.

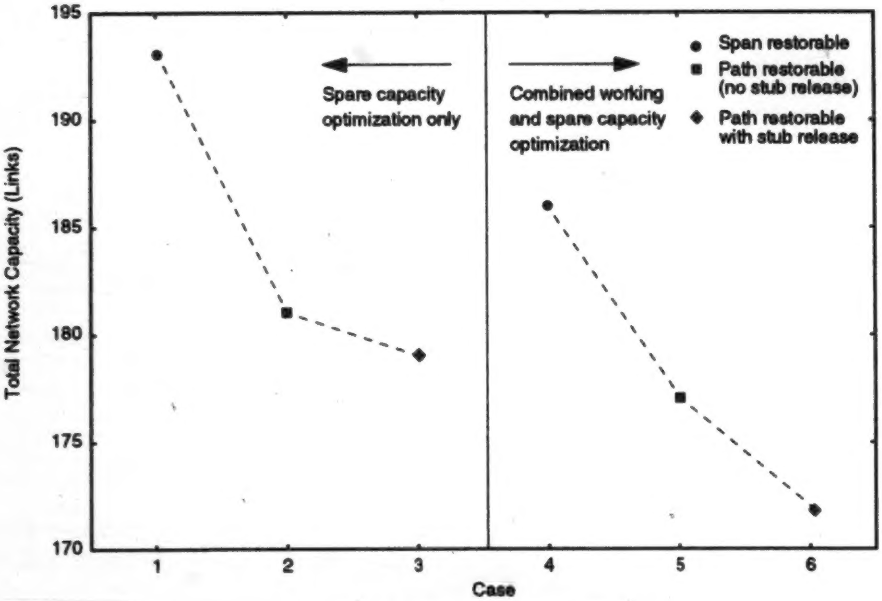


Figure 4.9. Network 1 Designs

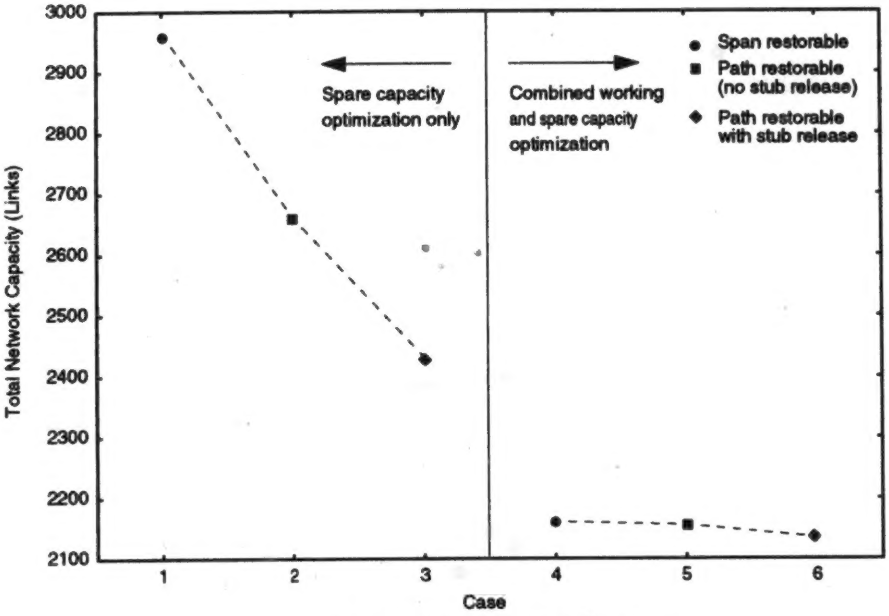


Figure 4.10. Network 2 Designs

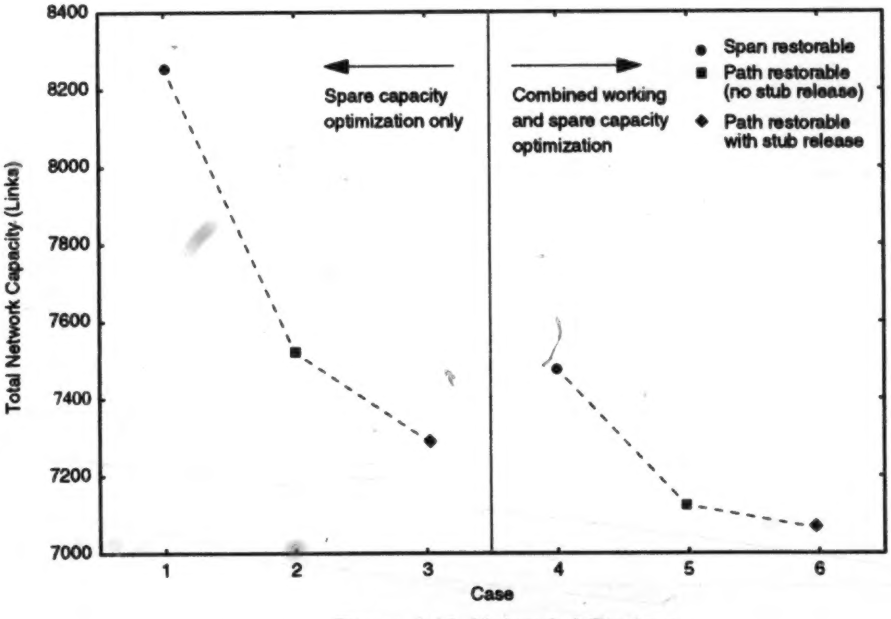


Figure 4.11. Network 3 Designs

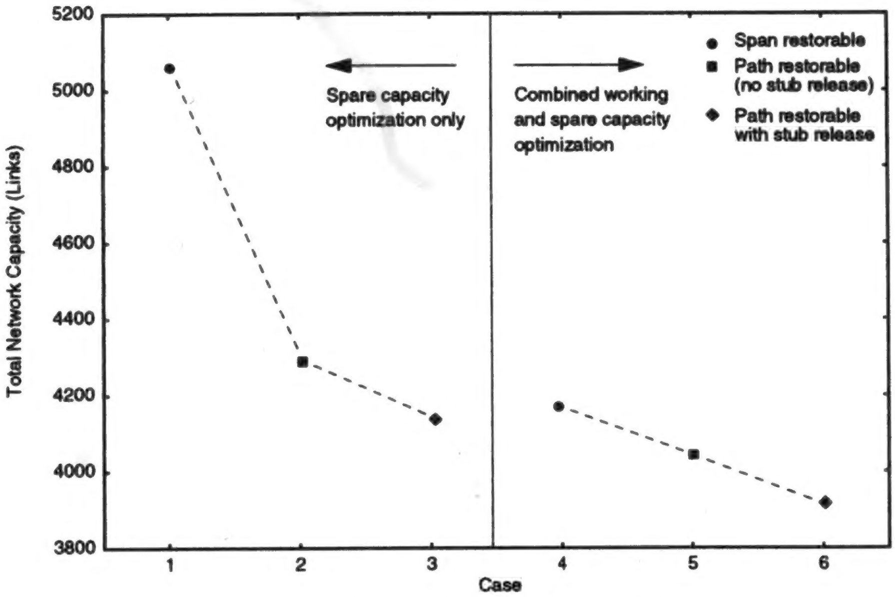


Figure 4.12. Network 4 Designs

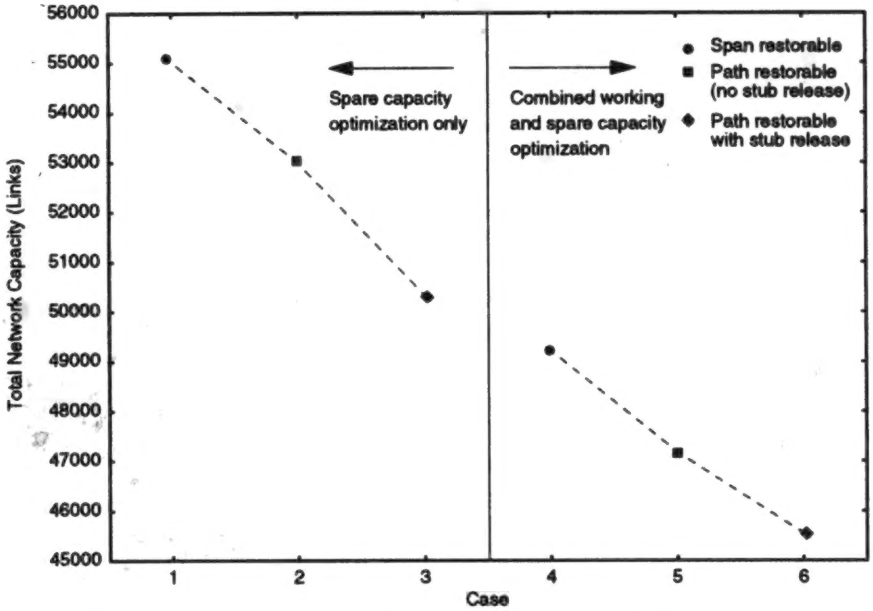


Figure 4.13. Network 5 Designs

Tables 4.2 through 4.7 presented next list the following information for each of the five networks investigated:

1. *Excess Distance Factor*: This factor specifies how much longer than the shortest route, in terms of physical distance, a potential restoration or working route between a demand pair may be.
2. *Excess Hop Factor*: This factor specifies how much longer than the shortest route, in terms of logical hop count, a potential restoration or working route between a demand pair may be.
3. *No. of eligible restoration routes*: This entry specifies the total number of potential routes available to the IP given the set of all distinct routes which satisfy the excess hop and distance factors, plus all routes in the set of k-successively shortest disjoint routes between all demand pairs not included in this set.
4. *No. of constraints*: This entry specifies the total number of constraints in the IP formulation, i.e. the number of rows in the IP tableau.
5. *No. of variables*: This entry specifies the total number of variables in the IP formulation, i.e. the number of columns in the IP tableau.

Table 4.2. Capacity Design Case 1

| Network | Excess Distance Factor | Excess Hop Factor | No. of eligible restoration routes | No. of constraints | No. of variables |
|---|---|---|---|---|---|
| 1 | ∞ | ∞ | 6 360 | 485 | 6 382 |
| 2 | ∞ | ∞ | 9 901 | 785 | 9 929 |
| 3 | ∞ | ∞ | 5 327 | 962 | 5 358 |
| 4 | 2 211 | 15 | 10 002 | 1 435 | 10 063 |
| 5 | 560 | 5 | 10 005 | 2 497 | 10 064 |

Table 4.3. Capacity Design Case 2

| Network | Excess Distance Factor | Excess Hop Factor | No. of eligible restoration routes | No. of constraints | No. of variables |
|---|---|---|---|---|---|
| 1 | 3 | 3 | 7 526 | 570 | 7 549 |
| 2 | 53 | 4 | 9 978 | 783 | 19 253 |
| 3 | 261 | 3 | 10 018 | 1 274 | 10 047 |
| 4 | 155 | 1 | 10 021 | 3 954 | 10 000 |
| 5 | 90 | 1 | 10 071 | 3 076 | 10 121 |

Table 4.4. Capacity Design Case 3

| Network | Excess Distance Factor | Excess Hop Factor | No. of eligible restoration routes | No. of constraints | No. of variables |
|---|---|---|---|---|---|
| 1 | 3 | 3 | 7 526 | 570 | 75 49 |
| 2 | 53 | 4 | 9 978 | 783 | 19 253 |
| 3 | 261 | 3 | 10 018 | 1 274 | 10 047 |
| 4 | 155 | 1 | 10 021 | 3 954 | 10 000 |
| 5 | 90 | 1 | 10 071 | 3 076 | 10 121 |

Table 4.5. Capacity Design Case 4

| Network | Working route excess distance factor | Working route excess hop factor | Res. route excess distance factor | Res. route excess hop factor | No. of eligible working & restoration routes | No. of constraints | No. of variables |
|---|---|---|---|---|---|---|---|
| 1 | ∞ | ∞ | ∞ | ∞ | 21 097 | 552 | 21 142 |
| 2 | 68 | 6 | ∞ | ∞ | 20 677 | 880 | 20 733 |
| 3 | 530 | 7 | ∞ | ∞ | 20 641 | 1 146 | 20 703 |
| 4 | 0 | 0 | 2 505 | 16 | 20 516 | 2 128 | 22 802 |
| 5 | 0 | 0 | 434 | 6 | 20 532 | 3 126 | 20 650 |

Table 6. Capacity Design Case 5

| Network | Working route excess distance factor | Working route excess hop factor | Res. route excess distance factor | Res. route excess hop factor | No. of eligible working & restoration routes | No. of constraints | No. of variables |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 4 | 4 | 36 260 | 956 | 19 253 |
| 2 | 0 | 0 | 30 | 4 | 21 094 | 1 478 | 21 120 |
| 3 | 0 | 0 | 230 | 2 | 20 492 | 2 679 | 23 233 |
| 4 | 0 | 0 | 37 | 0 | 20 450 | 9 303 | 20 608 |
| 5 | 0 | 0 | 0 | 0 | 20 566 | 8 122 | 20 684 |

Table 7. Capacity Design Case 6

| Network | Working route excess distance factor | Working route excess hop factor | Res. route excess distance factor | Res. route excess hop factor | No. of eligible working & restoration routes | No. of constraints | No. of variables |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 4 | 4 | 36 260 | 956 | 19 253 |
| 2 | 0 | 0 | 30 | 4 | 21 094 | 1 478 | 21 120 |
| 3 | 0 | 0 | 230 | 2 | 20 492 | 2 679 | 23 233 |
| 4 | 0 | 0 | 37 | 0 | 20 450 | 9 303 | 20 608 |
| 5 | 0 | 0 | 0 | 0 | 20 566 | 8 122 | 20 684 |

For each network topology and capacity placement technique the number of spare and working links per span, the physical length of each span, the nodes terminating a span, and the physical and logical redundancy of the network are specified in [38]. The design for network number 1, capacity design case number 1 from [38], is shown below as an example.

| Span | NodeA | NodeB | Distance | Working | Spare |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1.000000 | 8 | 3 |
| 2 | 0 | 2 | 1.000000 | 7 | 4 |
| 3 | 0 | 3 | 1.000000 | 7 | 4 |
| 4 | 1 | 3 | 1.000000 | 5 | 3 |
| 5 | 1 | 4 | 1.000000 | 6 | 4 |
| 6 | 1 | 5 | 1.000000 | 9 | 1 |
| 7 | 2 | 3 | 1.000000 | 7 | 3 |
| 8 | 3 | 6 | 1.000000 | 4 | 0 |
| 9 | 3 | 7 | 1.000000 | 9 | 1 |
| 10 | 3 | 4 | 1.000000 | 6 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| 11 | 4 | 6 | 1.000000 | 7 | 1 |
| 12 | 4 | 7 | 1.000000 | 5 | 1 |
| 13 | 4 | 5 | 1.000000 | 6 | 4 |
| 14 | 2 | 8 | 1.000000 | 10 | 0 |
| 15 | 2 | 6 | 1.000000 | 4 | 4 |
| 16 | 6 | 8 | 1.000000 | 5 | 3 |
| 17 | 6 | 7 | 1.000000 | 4 | 0 |
| 18 | 7 | 8 | 1.000000 | 7 | 3 |
| 19 | 7 | 9 | 1.000000 | 7 | 3 |
| 20 | 5 | 7 | 1.000000 | 6 | 2 |
| 21 | 8 | 9 | 1.000000 | 6 | 4 |
| 22 | 5 | 9 | 1.000000 | 7 | 3 |

Total Number of working links = 142
Total Number of spare links = 51
Total Number of working and spare links = 193
Physical Redundancy = 0.36
Logical Redundancy = 0.36
Average Network Nodal Degree = 4.40

## 4.5 Discussion of Capacity Placement Results

### 4.5.1 Capacity Savings of Path Restoration in Networks with Pre-defined Demand Routing

To quantify the capacity benefits of path restoration in networks with established working routes, the total capacity required in cases 2, and 3 was normalized to that of the comparable span restorable design (case 1). Figure 4.14 shows that when the placement of spare capacity is optimized, path restorable designs *without* stub release require between 4% and 15% less *total* capacity than the span restorable designs. The corresponding reductions in *spare* capacity investment were from 5% to 19%.

Results also show that stub release can further reduce the total capacity required in a path restorable network between 1% and 8%, as seen in Figure 4.14. Whether this further savings is worthwhile in practice may depend on the extent to which stub release complicates reversion to the original pre-failure state. In large networks the economic benefits of stub release may be substantial considering that stub release decreased the total number of links required in network 5 by 2695 (Figure 4.13, case 2 vs case 3).

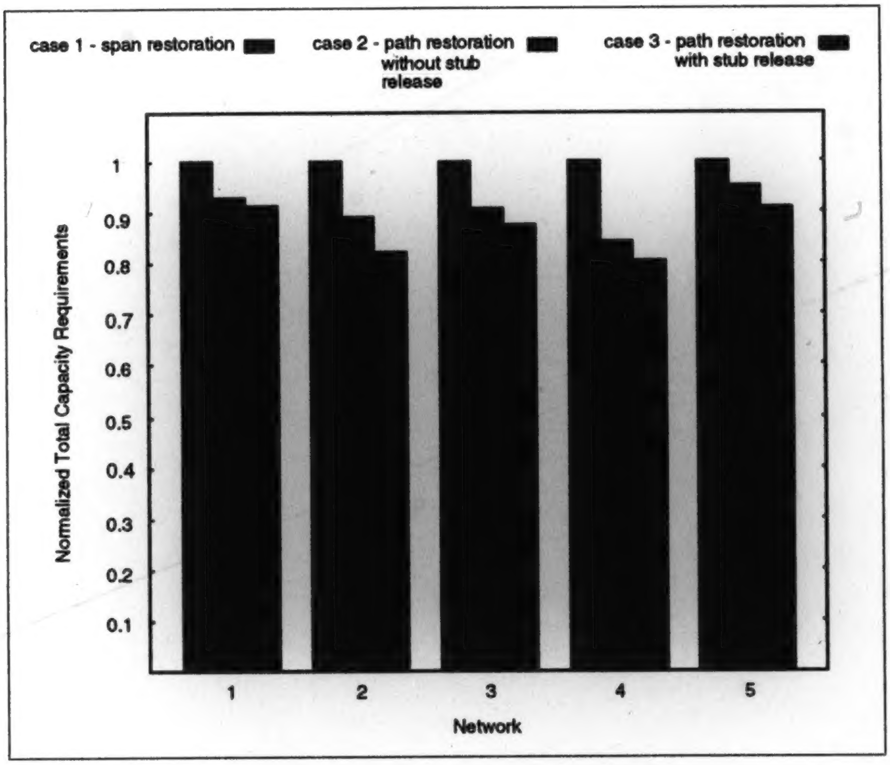


Figure 4.14. Optimization of Spare Capacity

### 4.5.2 Capacity Savings due to Joint Working and Spare Optimization

The total capacity savings gained when working path routing and spare capacity placement are jointly optimized is seen in Figures 4.9 - 4.13 when cases 1 - 3 are compared to cases 4 - 6. The benefit of combined optimization ranges from a total capacity reduction of 4% to 27% in the span restorable designs (comparing case 1 and case 4 results), averages 8% in the path restorable designs without stub release (comparing case 2 and case 5 results), and is about 7% in the path restorable designs with stub release (comparing case 3 and case 6 results).

Combined working and spare capacity optimization requires approximately double the number of routes to define its constraint system relative to those required for spare capacity optimization alone. This is because eligible working routes need to be specified in addition to eligible restoration routes. Given that the set of eligible routes is much larger when determining a combined capacity design, the constraint systems for cases 4, 5, and 6, include a smaller proportion of all possible routes than the constraint systems used to optimize the placement of spare capacity. It was only possible to include *all* routes in the constraint set and find the *global* optimum *spare* capacity placement for networks 1, 2, and 3 in a *span* restorable design (case 1).

Similarly, the constraint set for all path restorable designs included a smaller proportion of all possible routes than the constraint set for all span restorable designs. Path restoration needs to consider eligible working and restoration routes for all node pairs in the demand matrix, unlike span restoration which only needs to consider restoration routes between the nodes terminating a cut span. As a result, even though approximately the same total number of restoration routes were used to define the constraint system in each case, the jointly optimized designs included a smaller fraction of all possible routes when compared to the spare capacity designs, and the path restorable designs included a smaller fraction of all possible routes when compared to the span restorable designs.

A design is more likely to be near the global optimum when the IP is presented a proportionally larger area of the solution space. Therefore, the path restorable and combined capacity designs here may not be as close to their optimal capacity placement as the span restorable and spare capacity designs are to their global optimum, i.e. the benefits for cases 5 and 6 (especially) may be even greater than reported here. However, the theoretical benefits are not expected to be much greater than those reported here because of the large size of the eligible route set used in each case and the relatively small number of routes from that set used by the IP to restore a failure. For example, of those eligible routes specified for network number one in Tables 4.2, 4.4, and 4.5, a total of 105, 125, and 187 routes are used by the IP in cases 1, 3, and 4 respectively. It therefore seems plausible that the eligible route set formulated in each case is large enough to contain those routes which the IP needs to find the optimal capacity placement.

### 4.5.3 Routing Effects of Joint Working and Spare Optimization

The jointly optimized designs have some interesting properties which upon closer examination reveal characteristics associated with demand routing dispersion from the bounding considerations presented at the beginning of this chapter. When the IP is allowed to jointly optimize the placement of working and spare capacity in a network, it will choose working paths which are coordinated with the network restoration process. In span restorable networks this means that demands may sometimes be routed via paths longer than the shortest path. Table 4.8 indicates that one outcome of joint optimization is relatively evenly loaded span working quantities. This probably arises because an even distribution of span working quantities tends to prevent a single span failure from significantly raising the spare capacity requirements of the network. With the latitude to seek longer working routes, the IP does so to gain this benefit.

Table 4.8. Standard deviation in span working capacity confirming demand routing dispersion effect

| Network | Standard deviation of $w_j$ in case 1 designs | Standard deviation of $w_j$ in case 4 designs |
|---|---|---|
| 1 | 1.59 | 0.891 |
| 2 | 68.2 | 34.6 |
| 3 | 115 | 97.5 |
| 4 | 27.3 | 22.2 |
| 5 | 617 | 536 |

As noted in section 4.1, in path restorable networks it is often advantageous to disperse point-to-point demands over working paths such that the impact of a span failure affects many relations, even when such paths are slightly longer than the shortest route. Spreading the impact of a failure over a large region of the network increases the alternatives available when optimizing capacity placement and lowers the theoretical bound on redundancy for path restoration. Longer working paths also increase the amount of capacity available for stub release. However, the aim when designing a restorable network is not only to minimize the spare capacity, but to minimize the sum of the working and spare links required. Because longer working paths require more links, it is reasonable to expect that the IP design should have working paths which deviate only slightly from the shortest route. Indeed, Table 4.9, confirms that the average working path length in the path restorable designs is longer than the average working path length in the span restorable designs, but only marginally so. This suggests that the IP is able to achieve significant route dispersion of demands (Table 4.8) without actually increasing route lengths much in doing so (Table 4.9). Furthermore, Table 4.9 in conjunction with Figures 4.9 - 4.13 reveals that in all cases path restoration with stub release benefited the most from a tendency to allow longer working paths in the optimal design.

Table 4.9. Average Working Path Lengths

| Network | Shortest path routing of demands (Design cases 1, 2, and 3) | Jointly optimized designs | | |
|---|---|---|---|---|
| | | Span restoration (Design case 4) | Path restoration (Design case 5) | Path restoration with stub release (Design case 6) |
| 1 | 1.6 km | 1.6 km | 1.6 km | 1.6 km |
| 2 | 15.5 km | 17.1 km | 16.7 km | 17.8 km |
| 3 | 109.4 km | 115.1 km | 118.0 km | 131.3 km |
| 4 | 818.5 km | 848.9 km | 839.4 km | 994.7 km |
| 5 | 123.3 km | 128.6 km | 133.1 km | 137.9 km |

Despite the advantages of combined working and spare optimization, deploying and operating jointly optimized networks may be difficult in practice because working routes might have to be rearranged in the face of changing demand to retain overall joint capacity optimality. Given that the working capacities of most transport networks have already been defined, it may be a practical decision to continue provisioning working capacity based on shortest path routing, and separately optimize the placement of spare capacity.

## 4.6 Summary of IP Network Designs

The following tables summarize the six capacity designs for the five networks shown in Figures 4.9 - 4.13, and the normalization values used to construct the plot shown in Figure 4.14.

Table 4.10. Summary of IP Network Designs

| Network | Total capacity requirements | | | | | |
|---|---|---|---|---|---|---|
| | Case 1 (links) | Case 2 (links) | Case 3 (links) | Case 4 (links) | Case 5 (links) | Case 6 (links) |
| 1 | 193 | 181 | 179 | 186 | 177 | 172 |
| 2 | 2 958 | 2 655 | 2 428 | 2 159 | 2 158 | 2 134 |
| 3 | 8 250 | 7 526 | 7 287 | 7 479 | 7 125 | 7 061 |
| 4 | 5 070 | 4 292 | 4 134 | 4 174 | 4 043 | 3 915 |
| 5 | 55 024 | 53 031 | 50 336 | 49 237 | 47 143 | 45 516 |

Table 4.11. Summary of Spare Capacity Designs Normalized to Case 1

| Network | Normalized spare capacity requirements | | |
|---|---|---|---|
| | Case 1 | Case 2 | Case 3 |
| 1 | 1.00 | 0.94 | 0.93 |
| 2 | 1.00 | 0.90 | 0.82 |
| 3 | 1.00 | 0.91 | 0.88 |
| 4 | 1.00 | 0.85 | 0.82 |
| 5 | 1.00 | 0.96 | 0.92 |

## 4.7 Conclusions

Mesh restorable networks using path restoration *with* stub release are the most capacity efficient. Optimizing the spare capacity placement in such networks against all single span failures eliminated up to 19% of the total capacity of the corresponding span restorable networks studied here (see Figure 4.14 case 1 vs. case 3). The benefits of path restoration are also apparent when one considers that the average redundancy of all path restorable designs which optimized the placement of spare capacity only was 66%, and 87% for all span restorable designs.

The capacity required in a 100% path restorable network with stub release can further be minimized by an average of 7% when jointly optimizing the placement of spare and working capacity. The benefits of jointly optimizing working path routing and spare capacity placement are more pronounced in span restorable designs. Comparing jointly optimized span restorable designs to the baseline of a spare-only optimized span restorable design, as much as 27% may be saved in terms of total network capacity.

These findings and the IP design method for path restorable networks are one of the main contributions of this work. The integer programming formulation used to solve the spare and combined capacity placement problem in this chapter is flexible enough to accommodate span or path restorable networks and stub release if desired. A completed IP run specifies not only the spare and working capacity per span, but also the corresponding routing of working and restoration paths. Though it is possible to use the restoration pathset information to achieve centralized restoration, it is used here instead to test OPRA's ability to efficiently restore a span cut as explained in the next chapter.

# PART II: Optimized Distributed Path Restoration Algorithm

The second part of this thesis focuses on an optimized distributed real time solution to the path restoration problem, named OPRA. OPRA is the result of the first thorough exploration, development, realization, and testing of a bidirectional distributed path restoration algorithm. Part two of this thesis begins by defining appropriate figures of merit through which the performance and efficiency of OPRA can be assessed systematically and quantitatively. Then the principle at the heart of OPRA which enables it to find a near optimal set of link disjoint replacement paths within two seconds is explained. OPRA itself is described in detail in chapter 8. The remaining chapters present the testbed in which OPRA was implemented for experimental characterization and results from various tests of OPRA.

# Chapter 5. Performance Metrics and Theoretical Objectives for OPRA

This chapter presents appropriate figures of merit through which the performance and efficiency of OPRA, or any other distributed path restoration scheme, can be assessed systematically and quantitatively. These figures of merit are the same as those presented in [25] for span restoration, but evaluated differently as explained in this chapter given that this work investigates path restoration. One performance measure of obvious interest is speed of restoration. Another is the restorability of each span and the network as a whole given only the theoretical minimum of spare capacity. Restorability is the key measure for evaluating the operational performance of a distributed restoration mechanism and the spare capacity design of the network in which the mechanism operates.

In addition to defining speed-related and restorability performance measures, which can be categorized as operational metrics, two intrinsic path metrics are also discussed in this chapter. Intrinsic path metrics assess the efficiency of a restoration mechanism by measuring the method's ability to make maximum use of the resources which it is given relative to an idealized reference behavioural model. The two intrinsic path metrics discussed in this chapter are the Path Number Efficiency (PNE) and Path Length Efficiency (PLE).

## 5.1 Operational Performance Metrics

In any real transport network each span has a finite number of working and spare circuits. When a span is cut in such a network, the prime concern is to restore all of the lost working capacity on the given span. The ability to do this depends on the amount and placement of spare capacity in the network, as well as on the method used to find the restoration paths. Full restoration of working circuits may or may not require all of the paths that are topologically possible between all of the relations affected by a span cut. This distinguishes the operational measures of performance from the measures which are of interest from a theoretical performance viewpoint. In the latter, one wishes to characterize the intrinsic ability of a method to find efficiently all the paths that are topologically possible.

### 5.1.1 Span Restorability and Network Restorability

In an operational context, the utmost concern is with the proportion of failed working capacity that is restored. The restorability (or restoration ratio) of an individual span, i, having $w_i$ working links is:

$$Rs_i = \frac{min(w_i, k_i)}{w_i} \qquad 5.1.1.1$$

where $k_i$ is the number of restoration paths feasible using the restoration mechanism deployed in the network. Whenever $k_i \geq w_i$, $Rs_i = 1$. The restorability of a network as a whole (also called the Network Restoration Ratio (NRR)) is:

$$Rn = \frac{\sum_{i=1}^{S}[min(w_i, k_i)]}{\sum_{i=1}^{S}[w_i]} = \frac{\sum_{i=1}^{S}[Rs_i \cdot w_i]}{\sum_{i=1}^{S}[w_i]} \qquad 5.1.1.2$$

where S is the number of spans in the network. Note that Rn is not the average of individual span restorabilities as ratios except if all spans have equal $w_i$. As defined, Rn weights the restorability of each span by the size of each span, so that it expresses the total fraction of working capacity that is protected, not the average fraction protected on each span. This reflects the impact large spans have on network restorability.

The worst case restorability of a network (to span failures) is defined as the lowest span restorability level of any span in the network:

$$Rn_{wc} = min(Rs_i) \qquad i \in S \qquad 5.1.1.3$$

In the case of $Rn < 1.0$, $Rn_{wc}$ is a lower bound on $Rs_i$.

### 5.1.2 Speed Related Performance Measures

When considering a restoration scheme, four speed-related figures of merit are of interest. The first three of these can be applied either to an individual span cut or to a network as a whole by averaging the corresponding measure for every span cut in the network:

*A. First path time $t_{p1}$*: This is the time required to complete the first path of a restoration plan after the receipt of the first alarm. Its importance is primarily to assess the impact of a failure on high priority demands, and to characterize the speed of a distributed restoration algorithm when it acts in a mode imitative of an APS system for single-link failures. One may arrange priority use of the first path(s) found in a restoration plan so as to minimize the outage for priority services. The first path time indicates the minimum interruption that a priority service would see and gives an immediate indication if any portion of the affected traffic was restored before the call dropping threshold.

*B. Complete restoration time $t_R$*: This is the time to complete the last path that can be realized for restoration of a given span-cut. "Complete" does not imply 100% restoration. Rather, it is the time at which the transient restoration event is complete, however the outcome.

*C. Individual and mean path outage times $t_{p,i}$, and $t_{p,avg}$*: $t_{p,i}$ is the individual outage experienced by the $i^{th}$ working path restored. When several working links fail simultaneously the path outage times are identical to the sequence of path completion times. But in general, the onset of individual link failures may be time-dispersed within the failure event. Then $t_{p,i}$ is the elapsed time from the failure of the $i^{th}$ path until its individual restoration.

$t_{p,avg}$ is the average of $t_{p,i}$ over all working links restored in a restoration event. This is not necessarily the restoration time $t_R$ divided by k, the number of paths found. Due to the high degree of parallelism in some distributed restoration algorithms, the restoration time for the whole event ($t_R$) can be virtually the same as the individual outage time for every path ($t_{p,i}$) when link propagation delays exceed nodal processing delays.

*D. 95 percentile restoration time, $t_{95}$*: This is defined for a whole network, based on the individual outage times of all working paths, over all span cuts of the network. This is the time by which 95% of all individual links are restored and is the value at which the cumulative distribution function (CDF) of $t_{p,i}$ equals 0.95:

$$\int_0^{t_{95}} t_{p,l}(x)\,dx = 0.95 \qquad 5.1.2.1$$

where $t_{p,l}(x)$ is the probability density function of individual path outage times. The rationale for a 95 percentile level for specifications and measurement of outage times is that such a measure is less influenced by single extreme events in a sample.

## 5.2 Intrinsic Path Metrics

Span, network, and worst case restorability all depend on both the network design and the restoration re-routing scheme. Considering satisfactory restoration may be achieved by a weak routing mechanism in a generously spared network[1], separate *intrinsic* performance measures of a distributed restoration mechanism are needed. To assess the intrinsic efficiency of a proposed restoration mechanism, the results found by the restoration algorithm under evaluation must be compared to an ideal reference solution. Intrinsic path metrics thereby measure the restoration mechanism's ability to make efficient use of a network's spare capacity.

The routing effectiveness of a distributed restoration algorithm (DRA) may be evaluated by saturating the available spare capacity in a network design with a "full stress" path forming exercise. One way to do this given a path DRA is to assume each span cut in a network severs an effectively infinite number of working paths from every demand pair, and comparing the resulting pathsets found by each path restoration mechanism being evaluated to a reference solution. This approach is analogous to comparing the routing effectiveness of span DRAs by assuming that an effectively infinite number of working links are severed given any span cut in a test network, and comparing the resulting pathsets found by the span restoration mechanism to a reference solution as explained in [28]. However, performing a full stress test on a path DRA by severing an infinite number of working paths from every demand pair means running long simulations that require large amounts of computer memory for even moderately large networks. Unlike span restoration, the number of node pairs affected by a span cut in a path restorable network cannot be limited to a single pair of nodes. Consequently, exposing a path restoration mechanism to maximal path-finding stress by severing an infinite number of working paths from every demand pair is impractical for realistic transport networks.

---

[1] Generously in this context means having considerably more spares in total than the minimum theoretically required for 100% restorability of every span.

The routing effectiveness of various path DRAs are compared in this thesis under conditions of maximal path-finding stress using very tightly spared 100% restorable networks with little or no excess spare capacity. A "full stress" path forming exercise is possible using such tightly spared networks because the DRA being evaluated will saturate the available spare capacity in them. The 30 capacity efficient network designs presented in chapter 4 with Rn = 100% have little or no excess spare capacity and are therefore suitable to evaluate the routing effectiveness of various distributed path restoration algorithms.

The six capacity designs explained in section 4.4 for the five networks shown in section 4.3 form a set of 30 capacity efficient network designs that will be used to evaluate the performance of OPRA. Each network's spare capacity as detailed in [38] is the minimum required to restore all single span failures by the corresponding restoration mechanism. Ten of these designs optimize the placement of capacity to facilitate span restoration. However, in this thesis the mechanism used to restore a failure will always perform path, not span, restoration. All network designs are 100% restorable if the restoration pathset specified in the solution of the IP is used to restore a span cut. Any restoration mechanism deployed in these networks must make extremely efficient use of a network's spare capacity in order to restore all of the working paths severed by a span cut. Though a restoration mechanism does not necessarily have to duplicate the pathset found by the IP, the aim is for it to make equally efficient use of a network's spare capacity in order to restore all span cuts. The 30 network designs produced in chapter 4 will serve as stringent tests of any restoration mechanism's ability to efficiently restore a span cut.

### 5.2.1 Path Number Efficiency (PNE)

The routing efficiency, loop-freedomness, and predictability of a path restoration algorithm can be evaluated by comparing the pathsets found by the distributed path restoration algorithm under test to the reference pathsets found by the IP presented in chapter 4. The reference restoration pathsets are extracted from the IP capacity design information using the variable $f_i^{r,p}$ to identify which routes in the set of eligible predefined routes have a non-zero flow. The variable $f_i^{r,p}$ identifies the restoration flow through the $p^{th}$ restoration route for demand pair r in the IP formulation presented in the previous chapter. Each non-zero unit of flow identifies one reference restoration path which follows that route specified by $f_i^{r,p}$, i.e. route p.

Using this reference restoration path set, Path Number Efficiency is defined as:

$$PNE = \frac{\sum_{l=1}^{S} k_{dra,l}}{\sum_{l=1}^{S} k_{ref,l}} \qquad 5.2.1.1$$

where $k_{dra,l}$ is the number of paths found for span-cut l by the distributed restoration algorithm (DRA) under test, and $k_{ref,l}$ is the number of paths for span-cut l in the reference pathset. Failure to realize the number of paths found by the reference solution appears as a PNE less than 1. PNE = 1 implies ideal topologically-limited performance equal to that of the reference, or IP solution in this case. PNE = 1 does not necessarily imply 100% restorability because the restoration ratio depends on both the working and sparing levels in the network. Conversely a 100% restoration ratio does not imply PNE = 1 for the routing mechanism. Rn = 100% implies a restoration mechanism can fully restore each span of a network. Rn = 100% does not necessarily imply a restoration mechanism achieves ideal topologically-limited performance equal to that of the reference or IP solution.

### 5.2.2 Path Length Efficiency (PLE)

In restoration it is desirable to minimize any additional signal path delay for continental-scale networks and to make efficient overall use of a network's spares. In the extreme, poor routing in terms of path length, or looping paths, may also cause a drop in path number, so PLE and PNE are not totally unrelated. PLE is defined as the total distance of all paths in a restoration plan relative to the total path length of the corresponding reference solution pathset. PLE is defined only on the total length of the pathset, allowing individual path details to differ in two restoration plans which may be equivalent in terms of overall length efficiency. The path length efficiency of a distributed restoration algorithm in a given test network is:

$$PLE \equiv \frac{\sum_{i=1}^{S}\left[\sum_{j=1}^{k_i} L(P_{ref,i,j})\right]}{\sum_{i=1}^{S}\left[\sum_{j=1}^{k_i} L(P_{dra,i,j})\right]} \qquad 5.2.2.1$$

where L(P) is the length of path P, $P_{ref,i,j}$ is the $j^{th}$ path in the restoration plan for failed span i from the reference solution and $P_{dra,i,j}$ is the $j^{th}$ path in the distributed algorithm's restoration plan for the failed span i. k is the number of paths in the restoration plan identified by the distributed path algorithm or the reference solution. The PLE should not be used when the PNE of the reference solution and the distributed algorithm are not identical, because it is only meaningful to compare pathset length between solutions that have the same number of restoration paths. Note also that PLE is defined such that a path length total longer than the reference solution produces a PLE less than 1, (i.e. PLE = 1 is ideal).

## 5.3 Summary

The figures of merit presented in this chapter are the same as those presented in [25], and are necessary to systematically and quantitatively assess OPRA's performance in a manner that decouples its fundamental routing efficiency from exact levels of sparing in a network, just as they were necessary in [25] to systematically and quantitatively assess the performance of a span restoration mechanism, i.e. the SHN. The suite of network capacity designs presented in chapter 4 are to be used to test OPRA's ability to restore a network failure. Collectively the chapters to this point define a test-bed for any restoration mechanism, and establish the framework necessary to facilitate the detailed description of OPRA in the following chapters.

# Chapter 6. The Interference Heuristic for Coordinating Pathset Formation

The aim here in developing a distributed path restoration algorithm is to find a restoration pathset which maximizes Rn. Considering the number of permutations and combinations of paths possible when considering path restoration for even moderately large networks, it is often impossible to search the entire solution space for the optimal[1] restoration pathset in a practical amount of time, warranting the use of a heuristic to find a near-optimal pathset. This chapter presents the key heuristic at the heart of the proposed distributed path restoration algorithm, named OPRA, which enables it to find a collectively near-optimal set of link-disjoint replacement paths in potentially less than two seconds.

## 6.1 The Interference Principle

Most distributed restoration algorithms developed to date have focused on restoring span cuts before the two second call-dropping threshold in *span* restorable networks, or have implemented path restoration by methods equivalent to uncoordinated Capacity Scavenging. The distributed path restoration algorithm presented here, the Optimized Distributed Path Restoration Algorithm (OPRA), is unique in that it is the first algorithm which configures the surviving spare links of a *path* restorable network into a collectively near optimal multicommodity max-flow pathset. While optimizing capacity efficiency, OPRA maintains high restoration speed due to the parallelism inherent in its distributed implementation and its ability to synthesize a complete restoration pathset in a single pass. Unlike capacity scavenging, OPRA does not simply initiate multiple concurrent or sequential independent instances of a span restoration algorithm for each demand pair affected by a failure.

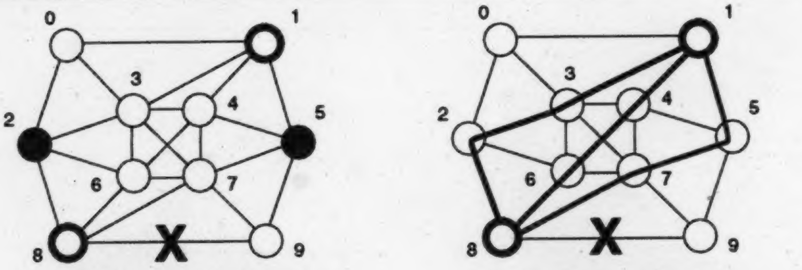
The heuristic used by OPRA to find a capacity-efficient restoration pathset is based on insights gained from extensive emulation and exploration of potential processes for path restoration. OPRA is the result of a systematic research methodology, as presented in this chapter, that began with extensive manual simulations. Initially path restoration was simulated in numerous test networks with Rn < 100% by tracing out on paper restoration paths through a network's spare capacity. Spare links would be concatenated together to form a restoration pathset and visually improved to restore as much lost working capacity as possible. Numerous simulations were performed in order to identify principles that led to the formation of a multi-commodity max-flow restoration pathset.

---

[1] The term optimal in part II of this thesis implies a restoration pathset which restores the maximum amount of lost demand topologically feasible, i.e. maximizes Rn, while satisfying constraints 1 through 4 defined in section 3.3.

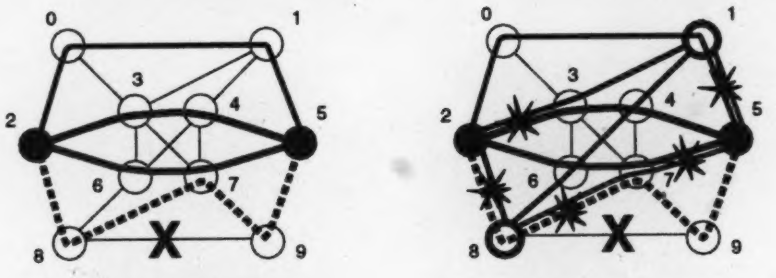
One of the main principles was found to be that given the set of all possible restoration paths, those paths which traverse spans with low spare capacity and/or nodes of low degree should be used last to maximize the restorability (Rn) of a network. Paths which traverse spans with low sparing and/or nodes of low degree render a large number of other restoration paths infeasible. Consequently OPRA prefers to use restoration paths which eliminate the fewest other restoration paths. This principle is referred to as the interference principle, and its distributed implementation in OPRA is the basis of the interference heuristic described later in this chapter.

Before proceeding directly to a distributed implementation of the interference principle, its effectiveness at maximizing Rn is verified. To do this, the "interference number" of a restoration path is defined. The interference number of a restoration path is defined as the number of other paths that a particular restoration path eliminates when all restoration paths synthesized in isolation by a restoration mechanism for each affected node pair are overlaid. The concept of a path's interference number is illustrated in Figure 6.1. Two relations are affected by the failure of the span between nodes 8 and 9. Given a network with a single spare link per span, Figure 6.1 shows the k-shortest link disjoint replacement paths for each affected demand pair in isolation. When the two sets of restoration paths are overlaid, it becomes clear that some paths are more "costly" than others in a global sense when evaluated in terms of the number of other restoration paths rendered infeasible by one path's existence. For example, restoration path 2 does not prevent the formation of any other path in either of the two pathsets, whereas implementation of restoration path 3 prevents formation of paths 4, 6, and 7. Path 3 is therefore assigned an interference number of three. In Figure 6.1 the overall restorability is maximized in the available sparing when the paths with the lowest interference numbers are chosen to restore the failure. Five units of demand can be restored using those restoration paths with the lowest interference number i.e. paths 2, 4, 5, 6, and 7, while only a maximum of 3 units of demand can be restored if restoration paths with the largest interference numbers, i.e. paths 1, and 3, are used.



(a) Network topology showing one spare link per span and demand pairs to be recovered

(b) k-shortest link disjoint restoration paths between nodes 1 and 8 in isolation



(c) k-shortest link disjoint restoration paths between nodes 2 and 5 in isolation

(d) interference between individual restoration pathsets

| Restoration Path | | Interference Number |
|---|---|---|
| —— | 1 | 2 (contention on spans 2-3 and 2-8) |
| —— | 2 | 0 |
| —— | 3 | 3 (contention on spans 1-5, 5-7, and 7-8) |
| —— | 4 | 1 (contention on span 1-5) |
| —— | 5 | 1 (contention on span 2-3) |
| —— | 6 | 1 (contention on span 5-7) |
| ▪▪▪▪▪▪▪▪ | 7 | 2 (contention on spans 2-8 and 7-8) |

Figure 6.1. Concept of Interference Numbers

Using restoration paths with low interference numbers to restore a failure inherently avoids creating and using spans and/or nodes which act as bottlenecks. By avoiding bottlenecks it is hypothesized that the number of restoration paths possible in a tightly spared network and the sharing of spare capacity between all relations seeking to restore lost demand is maximized. This hypothesis is confirmed by the results presented in the following section.

### 6.1.1 Validating the Interference Principle: A Centralized Path Restoration Algorithm

To test the use of interference number as a tactic to maximize network restorability (Rn) in a path restoration context, a conventional program was written which used those restoration paths with the lowest interference number to restore a failure. This program, named Interference Tester, is not the complete path restoration heuristic used in OPRA, and may find path interference numbers different from those shown in Figure 6.1 because a path's interference number depends on the path set used to evaluate it. Interference Tester was used as a centralized benchmark to approximate the pathset that would be found by a distributed interference heuristic, and to justify in advance the development of a true distributed restoration algorithm based on the local processing of interference numbers.

A flow chart of Interference Tester is shown in Figure 6.2. Initially Interference Tester identifies all node pairs affected by a failure. If stub release is being used, the surviving portions of cut working paths are added to the pool of network spare capacity. Next a composite set of eligible restoration paths, consisting of each affected node pair's k-successively shortest link-disjoint[1] paths found in isolation, is identified. Those restoration paths which traverse the end-nodes of a failed span are not included in the set of all eligible restoration paths when other feasible restoration paths exist for that demand pair. This is done to model a distributed restoration algorithm which restores paths that traverse nodes of low degree last, given that a span failure decreases the degree of the nodes terminating a cut span by one.

---

[1] Recall, in this work a link is only one unit of capacity on a span. This is not the same as span-disjoint paths.

From the set of all eligible restoration paths, the path with the lowest interference number is identified, and used to restore one unit of lost capacity. The interference number of a path is calculated by removing the spare capacity needed to support that path from the network and counting the number of other restoration paths rendered infeasible as a result. The spare capacity used by that restoration path is then returned to the network and the process repeated for the next eligible restoration path. This implementation of the interference principle only increments the value of a path's interference number when the remaining spare links on one or more span's the path traverses is equal to one.

If two or more restoration paths have the same minimum interference number, three options are available to resolve the tie, all of which are tested in the results presented subsequently. The first option selects the path associated with the relation which has lost the larger amount of demand, as to restore capacity in proportion to demand. The second option restores the relation with the fewest restoration paths. The last option selects the shorter restoration path, so as to use the least amount of spare capacity.

Spares used to complete a restoration path are removed from the network at each iteration when a unit of lost capacity is restored. When all of the lost capacity for a given span failure has been restored, or no feasible restoration paths remain, Interference Tester returns the networks spare capacity to its pre-failure values, and begins another iteration of the same steps described above for the next span cut, until the restorability of all spans in the network has been evaluated.

A completed Interference Tester run specifies a routing of restoration paths. Though it is possible to use this restoration pathset information to achieve centralized restoration, it is used here instead to test the use of the interference number as a tactic to maximize network restorability (Rn). If the results from Interference Tester, presented subsequently, indicate that the interference principle maximizes Rn, efforts to transform it into a heuristic which may be used by a distributed path DRA are warranted.

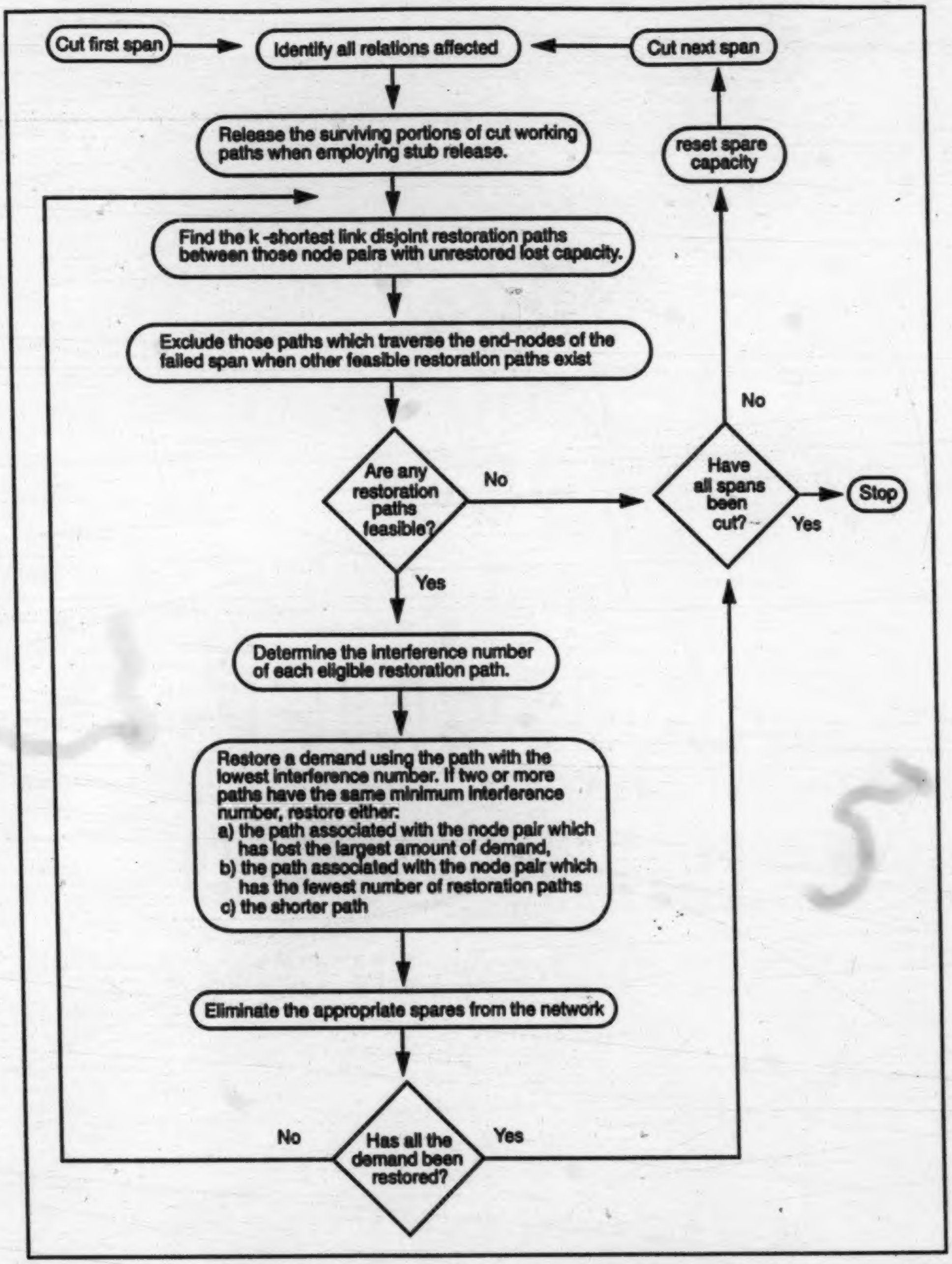

Cut first span
Identify all relations affected
Cut next span
Release the surviving portions of cut working paths when employing stub release.
reset spare capacity
Find the k -shortest link disjoint restoration paths between those node pairs with unrestored lost capacity.
Exclude those paths which traverse the end-nodes of the failed span when other feasible restoration paths exist
No
Are any restoration paths feasible?
No
Have all spans been cut?
Stop
Yes
Yes
Determine the interference number of each eligible restoration path.
Restore a demand using the path with the lowest interference number. If two or more paths have the same minimum interference number, restore either:
a) the path associated with the node pair which has lost the largest amount of demand,
b) the path associated with the node pair which has the fewest number of restoration paths
c) the shorter path
Eliminate the appropriate spares from the network
No
Has all the demand been restored?
Yes

Figure 6.2. Program used to test the interference principle

The results from Interference Tester are presented in Table 6.1. Path restoration was simulated in three of the five networks presented in Chapter 4 to gain insights on the effectiveness of the interference principle at maximizing Rn. Network 1, the small test network, as well as networks 2 and 4, representative of a metropolitan and a long haul network respectively, were used in the investigation. Using the 100% path restorable network designs specified by the IP presented in chapter 4 as reference, the Path Number Efficiency (PNE) of a path restoration mechanism based on the interference principle is shown.

For comparison, the PNE of a path restoration mechanism based on uncoordinated sequenced Capacity Scavenging was also calculated. As explained in section 1.4.4, Capacity Scavenging is the use of a span restoration algorithm to find replacement paths between the source and destination of every demand pair affected by a failure either successively or by letting all span restoration instances execute concurrently. The set of k-successively shortest link disjoint paths between those demand pairs seeking to restore lost capacity was used to model the restoration pathset found by the span restoration algorithm used in Capacity Scavenging. Rather than a concurrent implementation, the Capacity Scavenging results presented in Table 6.1 correspond to an implementation that restores one demand pair at a time. Four different demand pair orderings were investigated. The relations to be recovered were arranged in increasing and decreasing order according to the geographical distance separating them, and the amount of demand lost.

Given that the average PNE of all the values presented in Table 6.1 for a restoration mechanism based on the interference principle as implemented in Interference Tester is 98.8% with a standard deviation of 1.97%, restoring those paths which traverse spans with low sparing, and/or tandem through nodes of low degree last, often maximizes the path restorability of a network. All three options for resolving ties were equally good, suggesting a path's interference number alone is sufficient to maximize Rn. Real networks will likely have somewhat more spare capacity than the tightly designed reference networks used to test the interference principle due to the provisioning interval and the modularity effects of transmission systems. Therefore, based on the results presented in Table 6.1, a restoration mechanism based on the interference principle will likely achieve 100% network restorability in practice.

Table 6.1. Performance Studies of a Centralized Path Restoration Mechanism based on the Interference Principle

| Network | Capacity Design Test Case | PNE of a path restoration mechanism using Capacity Scavenging | | | | PNE of a path restoration mechanism based on the interference principle | | | Average increase in PNE resulting from the interference principle |
|---|---|---|---|---|---|---|---|---|---|
| | | Order relations by decreasing distance between nodes | Order relations by increasing distance between nodes | Order relations by decreasing lost demand | Order relations by increasing lost demand | Option 1 | Option 2 | Option 3 | |
| 1 | spare capacity optimization, path restoration with no stub release | 94.4% | 100% | 99.3% | 98.6% | 100% | 99.3% | 100% | 1.7% |
| 1 | spare capacity optimization, path restoration with stub release | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 0% |
| 1 | combined capacity optimization path restoration with no stub release | 93.7% | 95.1% | 95.1% | 93.7% | 100% | 100% | 100% | 5.6% |
| 1 | combined capacity optimization; path restoration with stub release | 98.6% | 97.2% | 97.9% | 97.9% | 100% | 100% | 100% | 2.1% |
| 2 | spare capacity optimization, path restoration with no stub release | 92.1% | 92.9% | 94.6% | 92.6% | 94.7% | 94.9% | 95.1% | 1.8% |
| 2 | spare capacity optimization, path restoration with stub release | 93.2% | 89.7% | 90.8% | 93.5% | 92.2% | 95.3% | 96.2% | 2.8% |
| 2 | combined capacity optimization path restoration with no stub release | 97.9% | 98.8% | 99.3% | 97.4% | 99.9% | 99.3% | 99.9% | 1.3% |
| 2 | combined capacity optimization, path restoration with stub release | 99.4% | 96.5% | 96.5% | 99.7% | 97.4% | 100% | 99.4% | 0.9% |
| 4 | spare capacity optimization, path restoration with no stub release | 96.6% | 99.4% | 99.2% | 96.9% | 99.0% | 99.5% | 99.6% | 1.4% |
| 4 | spare capacity optimization, path restoration with stub release | 98.5% | 99.8% | 99.6% | 99.2% | 99.8% | 99.9% | 100% | 0.6% |
| 4 | combined capacity optimization path restoration with no stub release | 97.3% | 99.3% | 98.2% | 98.4% | 98.9% | 100% | 99.8% | 1.3% |
| 4 | combined capacity optimization, path restoration with stub release | 98.5% | 99.7% | 99.8% | 99.5% | 99.7% | 99.5% | 99.5% | 0.2% |

Given that the average PNE of all the values presented in Table 6.1 for Capacity Scavenging is 97.2% with a standard deviation of 2.75%, compared to 98.8% and 1.97% for the interference principle when implemented using a restoration path's interference number, a path restoration mechanism based on the interference principle is likely to perform better than a path restoration mechanism based on iterative and sequential Capacity Scavenging. In Table 6.1, the path restoration mechanism based on the interference principle finds on average 1.6% more restoration paths than Capacity Scavenging, and on average only 1.2% fewer restoration paths than topologically feasible.

As shown in Table 6.1 Capacity Scavenging does remarkably well in terms of PNE. However, only a distributed path restoration mechanism based on the interference principle promises to restore a failure before the two second call dropping threshold. Even though distributed span restoration algorithms are likely to restore a span failure within two seconds, their repeated application in sequenced Capacity Scavenging would increase the total restoration time beyond the call-dropping threshold. Given the importance of restoring a network failure quickly as explained in chapter 1, speed is the main reason for pursuing a distributed path restoration algorithm based on the interference principle.

It may be argued that pursuing such a path DRA is unwarranted given that Capacity Scavenging is itself a distributed path restoration algorithm. However, a path restoration mechanism based on iterative Capacity Scavenging, and not the interference principle requires some type of a preplan and/or centralized implementation to at least dispatch the affected demand pairs in the prescribed sequence and coordinate Capacity Scavenging events. The goal of the present research is to develop a distributed algorithm, and only concurrent execution of Capacity Scavenging for all affected pairs satisfies this goal.

While concurrent Capacity Scavenging could potentially restore a failure before the 2 second call dropping threshold without some type of preplan and/or centralized implementation, the restoration pathset resulting from concurrent Capacity Scavenging does not explicitly adhere to the interference principle and will likely not correspond with the pathset found using iterative Capacity Scavenging. These pathsets will likely not correspond because the ordered sequence in which demand pairs synthesize restoration paths in iterative Capacity Scavenging is not present in concurrent Capacity Scavenging by definition. Given that the pathsets from iterative and concurrent Capacity Scavenging will likely differ, the PNE from concurrent Capacity Scavenging likely won't correspond with the PNE values shown in Table 6.1.

It is difficult to characterize the restoration pathset concurrent Capacity Scavenging would find because of its uncoordinated nature, making it difficult to reliably predict concurrent Capacity Scavenging's PNE for all failures. While concurrent Capacity Scavenging may achieve a very high PNE in some cases, it may not in others, and engineering a restoration mechanism that may unpredictably perform poorly for certain span failures is undesirable.

In conclusion, the goal of achieving an optimal solution to the restoration problem and the results obtained with Interference Tester favour developing a distributed path restoration mechanism based on the interference principle. This principle led to the definition of a restoration path's interference number and is the basis of the interference heuristic described next. The following section begins developing the interference heuristic by explaining how the interference principle is adapted to a distributed implementation.

## 6.2 Distributed Implementation of a Path DRA based on the Interference Principle

Distributed restoration algorithms generally rely on some form of a flooding process, and in the case of single iteration pathset forming DRA's like the SHN [25] and OPRA, incorporate some form of selective rebroadcasting to find an optimal set of link disjoint replacement paths. Flooding is a routing strategy which requires no network information and works as follows: a signal is sent by a source node to each of its neighbours; at each neighbour the incoming signal is retransmitted on all outgoing links except for the link that it arrived on. The basic idea of flooding may be familiar from data networks where it is used to distribute updates on network configuration. However, flooding in a DRA is quite specialized relative to these general notions of single flooding.

Distributed restoration exploits flooding in a way that derives a complete set of link-disjoint non-looping paths between two nodes, ideally doing so in only one iteration.

The pathset must be collectively shortest and consistent with the exact quantities of spare links available on each span in the network.

At first glance it is tempting to assume that simple flooding addresses the distributed restoration problem. However, simple flooding will only solve the distributed restoration problem if one replacement path on one route is selected per iteration [28]. For a distributed restoration algorithm, such as OPRA, flooding must be significantly modified. While simple flooding does enumerate all distinct routes in a network, the output from a distributed restoration algorithm must be a set of mutually feasible paths, not just a list of all routes. Simple flooding will generate a huge number of distinct routes, many of which are not capacity consistent paths or have loops [28]. The challenge for a distributed restoration algorithm is to find the relatively small subset of mutually consistent paths within the large number of distinct routes generated by flooding. OPRA uses the interference principle as represented in the interference heuristic to guide and control the flooding process such that a near-optimal link-disjoint restoration pathset is found in a single flooding phase.

Using the interference principle in Interference Tester required determining a restoration path's interference number. However, a distributed path restoration algorithm does not have access to a database of potential restoration paths, which makes it impossible to adapt the interference principle to a distributed implementation by counting how many other restoration paths a given path eliminates, as this implies a global network view. By design the desired DRA will not have more than a local-node state view, and a distributed restoration mechanism requires that the interference principle be implemented using only local information at a node.

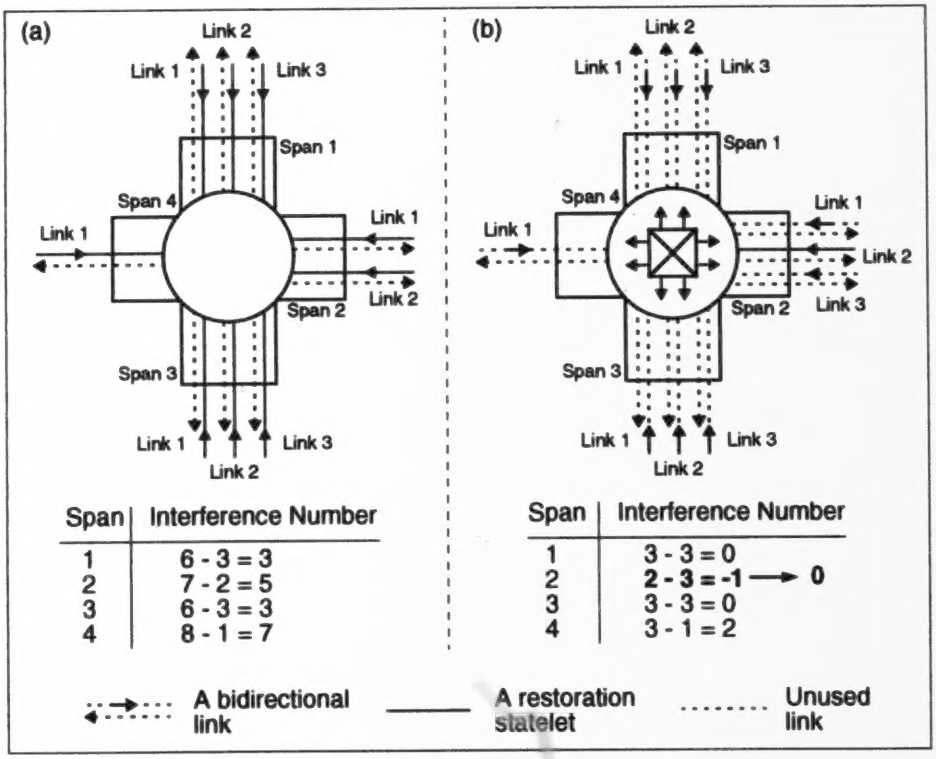
Shifting from a global to a node-local view requires associating interference numbers with the state based signals used to establish the framework for interaction between nodes in a DRA. These state based signals are referred to as "signatures" in the SHN [25], and the opportunity is taken now to both revise the name signature into restoration statelet, both to be slightly more descriptive in going forward in this new work, and to distinguish an SHN signature from an OPRA "signature" because of the new fields present in the latter. (The fields of a restoration statelet are described in detail in the following chapter.)

In the node-local view, the basic target broadcast pattern at a tandem node executing OPRA aims to forward one restoration statelet on one link on all spans it terminates except the span on which that restoration statelet arrived. The basic target broadcast pattern at a source node executing OPRA aims to forward on each span a number of restoration statelets equal to the number of working paths lost by this node. These basic broadcast patterns are similar to those presented in the SHN in [25] because they ensure many diverse restoration paths are evaluated when synthesizing the optimal restoration pathset. While the basic target broadcast patterns in the SHN and OPRA are similar, the rules which mediate the competition between statelets for outgoing spare links at a node are substantially different.

Often the target broadcast pattern at a node cannot be satisfied fully for each incoming restoration statelet because a span can only support a limited number of restoration statelets, equal to the number of spare links it contains. The interference principle is used to mediate the competition between restoration statelets for rebroadcast. Applying the interference principle in a node local context requires that those restoration statelets which lead to the formation of restoration paths that traverse spans with low sparing be deferred. Adapting the interference principle to a distributed implementation requires calculating a span's interference number, rather than a path's interference number. If spans with low sparing are assigned a high interference number, and spans with high sparing are assigned a low interference number, and restoration statelets traversing spans with low interference numbers are preferentially rebroadcast first at a node, a distributed implementation of the interference principle is achieved. This distributed implementation defines the interference heuristic.

The number of other restoration statelets that cannot obtain rebroadcast in a given span determine that span's interference number. The interference number of a span is calculated by counting the number of restoration statelets competing to be forwarded on a span and subtracting from this sum the number of statelets the span can support.

As shown in Figure 6.3 (a), two restoration statelets on span 2, three restoration statelets on span 3, and one restoration statelet on span 4, want to access the three spare links available on span 1, resulting in a span interference number of three for span 1. Likewise, restoration statelets originating at a node must compete with incoming signals for available spares. As shown in Figure 6.3 (b), the node wants to transmit two restoration statelets on each span it terminates and also satisfy the single restoration statelet received on span 2, resulting in an interference number of zero for span 1. (Dealing with negative interference number is discussed later in this chapter.) Notice that the interference number of a span remains constant while the target broadcast pattern is fulfilled at a node because the number of restoration statelets which would like to be forwarded on a span and the number of spare links available on that span for rebroadcast decrease monotonically in unison.


(a)
Link 2
Link 1
Link 3
Span 1
Span 4
Link 1
Link 1
Link 2
Span 2
Span 3
Link 1
Link 3
Link 2
(b)
Link 2
Link 1
Link 3
Span 1
Span 4
Link 1
Link 1
Link 2
Link 3
Span 2
Span 3
Link 1
Link 3
Link 2

(a)

| Span | Interference Number |
|---|---|
| 1 | 6 - 3 = 3 |
| 2 | 7 - 2 = 5 |
| 3 | 6 - 3 = 3 |
| 4 | 8 - 1 = 7 |

(b)

| Span | Interference Number |
|---|---|
| 1 | 3 - 3 = 0 |
| 2 | **2 - 3 = -1 → 0** |
| 3 | 3 - 3 = 0 |
| 4 | 3 - 1 = 2 |

A bidirectional link
A restoration statelet
Unused link

Figure 6.3. Span Interference Numbers

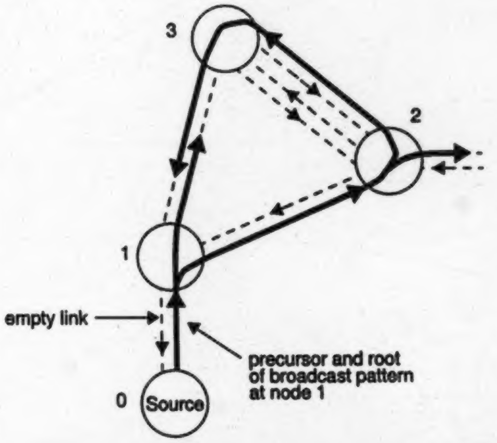
During restoration the interference number of a span will change as restoration statelets appear and disappear at a node. Calculating the interference number of a span associates larger values with spans that are in high demand and have little spare capacity, and smaller values with spans that are in low demand and have a lot of spare capacity.

The interference number of each span a restoration statelet traverses is added to the value of that statelet's interference number field. Initially the value of a statelet's interference number field is zero at the source node for a given demand pair, and accumulates as a statelet is rebroadcast at tandem nodes. The interference number of a path can then be estimated from the sum of the interference numbers of all the spans a statelet traverses. As verified by the results presented at the end of this thesis, the interference principle implemented at the node level using the interference heuristic, which rebroadcasts those restoration statelets with the lowest interference number first, results in a near optimal restoration pathset at the network scale because paths which traverse spans with low sparing are not used unless no other restoration paths remain.

When there are fewer spares on a span than there are restoration statelets wanting to obtain rebroadcast through it, a span's interference number is a positive integer. When there are more spares on a span than there are restoration statelets wanting to access it, a span's interference number is zero. A span is never allowed to have a negative interference number, as shown in Figure 6.3 (b), because negative interference numbers can lead to looping restoration paths.

If spans are allowed to have negative interference numbers, it is possible that a statelet travelling a cyclical path will have a lower interference number than its predecessor, identified as the precursor as in Figure 6.4, at the node completing the cycle (node 1 in Figure 6.4). If the interference number of the statelet arriving at node 1 in Figure 6.4 is less than that of its precursor, OPRA would shift the root of the broadcast pattern at node 1 from the span between the source and node 1 to the span between nodes 1 and 3, forming a loop in which the statelet may cycle forever.

When spans are prevented from having negative interference numbers, a statelet which has travelled a cyclical path never has a lower interference number than its precursor. It may have the same interference number if all the spans it traverses have an interference number of zero, but since statelet arrivals which are not of the lowest interference number for that family are ignored by OPRA, looping paths are prevented. (Details of the rules OPRA follows when processing statelets are detailed in chapter 8 and only mentioned here to explain the need for positive span interference numbers.)

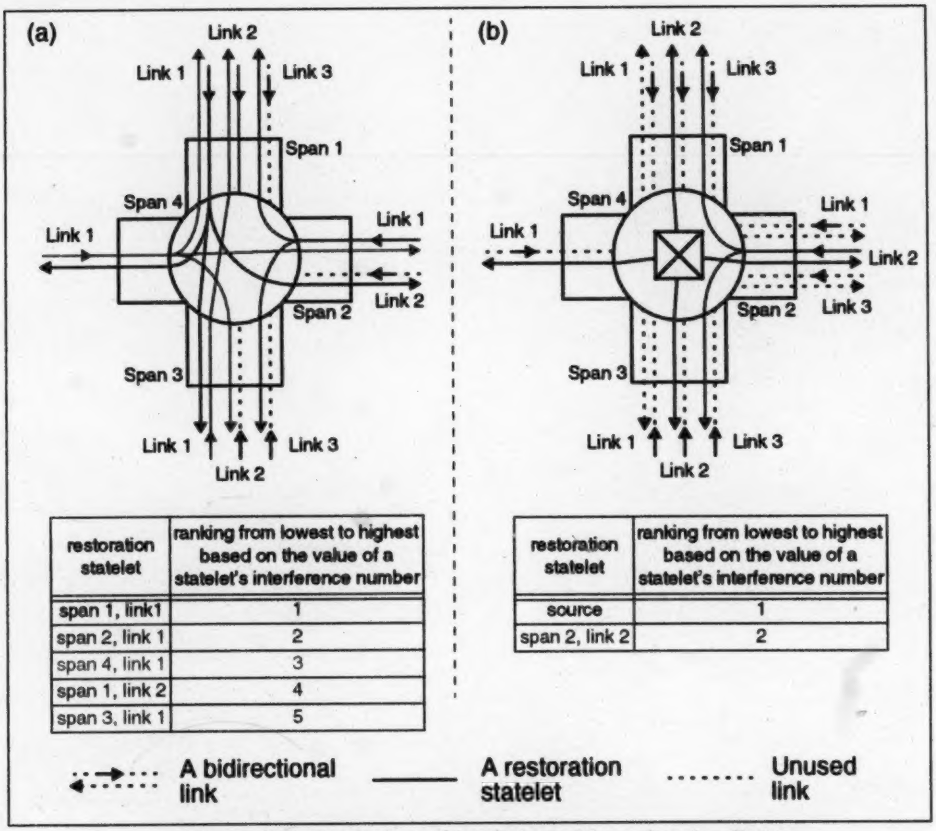


| Span | Interference number of span | Interference number of statelet |
|---|---|---|
| 0 - 1 | 0 | 0 |
| 1 - 2 | 0 | 0 |
| 2 - 3 | -1 | -1 |
| [illegible] | 0 | -1 |

Figure 6.4. Generating a cyclical path using negative span interference numbers

Limiting the interference number of a span to zero prevents those spans which have a lot of spare capacity from attracting statelets. In a sense a positive interference number may be viewed as a warning to statelets that traversing a span is "expensive", and a negative interference number may be viewed as an advertisement to statelets that traversing a span is "cheap". However, setting a lower limit of zero on the interference number of a span does not compromise OPRA's ability to synthesize a near-optimal restoration pathset because a span's interference number is only zero when an abundance of spares exist in the network. In a generously spared network, finding a pathset that achieves 100% restoration is simple. When sparing is scarce and it is critical to optimize the restoration pathset, span interference numbers are generally greater than zero and OPRA is more effective in finding restoration paths.

An example of the basic broadcast pattern which results at the node level when the interference heuristic is followed and those restoration statelets with the lowest interference number are rebroadcast first at a tandem node is shown in Figure 6.5 (a). Figure 6.5 (b) shows a similar situation when a node simultaneously rebroadcasts/tandems and sources restoration statelets. The broadcast patterns shown in Figure 6.5 assume each statelet belongs to a different demand pair, and that each demand pair lost one unit of demand. The broadcast patterns shown aim to forward one restoration statelet on one link on all spans the node terminates, except the span on which a restoration statelet arrived in the case of a tandem node. The broadcast pattern for each statelet is either fully satisfied in Figure 6.5 (i.e. has one rebroadcast in every span) or is partially satisfied to the greatest extent possible, consistent with a statelet's overall rank in terms of its interference number and the relative spans in which it and other statelets lie. The structure of this broadcast pattern is a result of the interference heuristic and explained in more detail in chapter 8, including any changes resulting when statelets belong to the same demand pair.


(a)
Link 2
Link 1
Link 3
Span 1
Span 4
Link 1
Link 1
Link 2
Span 2
Span 3
Link 1
Link 3
Link 2
(b)
Link 2
Link 1
Link 3
Span 1
Span 4
Link 1
Link 1
Link 2
Link 3
Span 2
Span 3
Link 1
Link 3
Link 2
A bidirectional link
A restoration statelet
Unused link

| restoration statelet | ranking from lowest to highest based on the value of a statelet's interference number |
|---|---|
| span 1, link1 | 1 |
| span 2, link 1 | 2 |
| span 4, link 1 | 3 |
| span 1, link 2 | 4 |
| span 3, link 1 | 5 |

| restoration statelet | ranking from lowest to highest based on the value of a statelet's interference number |
|---|---|
| source | 1 |
| span 2, link 2 | 2 |

Figure 6.5. Examples of basic target broadcast patterns

As shown in Figure 6.5 (b), a distinction between the SHN [25] and OPRA is that a single node executing OPRA may simultaneously initiate transmitting restoration statelets, a function performed by a node when acting as a source, and rebroadcast statelets from other demand pairs, a function associated with a tandem node. It may also receive restoration statelets from the far-end node of a failed working path terminated at that node, a function associated with a destination. In contrast, a single node in the SHN functions as either a Sender, Chooser, or tandem node for a given "fault instance", but never as a Sender, Chooser, or tandem node simultaneously. Given that a single node executing OPRA must be able to perform three roles simultaneously, the terms Sender and Chooser from the SHN, which explicitly associate one role with a single node, are avoided when describing OPRA. Instead, the functions performed by a node classify whether it is acting as a source, destination, or tandem node for the statelet being processed.

The interference number of a span quantifies the competition between restoration statelets for spares at a node, similar to the way the interference number of a path quantifies the competition for spare capacity in a network in the prior centralized view. Ultimately the interaction and competition between statelets results in a restoration path as explained in subsequent chapters.

Estimating the interference number of a path based on span interference numbers, which may be different from the interference number of that path as calculated using Interference Tester, lends itself to a distributed implementation. All nodes in a network can simultaneously and independently calculate a span's interference number as restoration statelets appear and disappear. Furthermore, the interference number of a span can be added to the interference number of a restoration statelet before it is transmitted on that span, so that the interference number of a path accumulates as the restoration statelet transits a network. When restoration statelets traversing spans with low interference numbers are preferentially rebroadcast first at a node, a distributed implementation of the interference principle is achieved. This distributed implementation of the interference principle is called the interference heuristic.

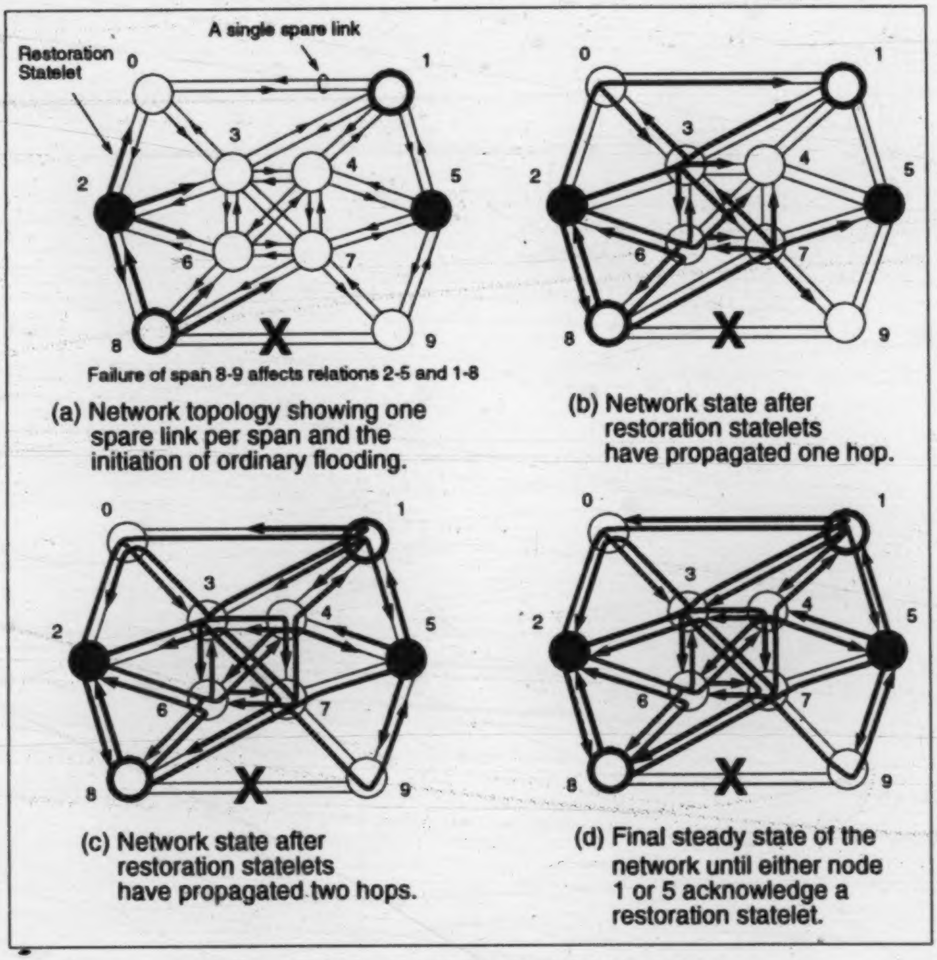
### 6.2.1 End-node Bottleneck Effect and the need for Bidirectional Flooding

While the interference number of a span can be used by a restoration algorithm to defer using paths which traverse spans with relatively few spare links, it cannot be used to defer using paths which traverse nodes of low degree. It is especially advantageous to defer selecting restoration paths which tandem through the source or destination of lost demands. A strong principle identified while performing manual path restoration simulations is that the spare links incident with the source and destination of a failed working path are most efficiently used to restore the demand originating at that node because two spare links can then be used to restore two units of lost demand instead of one.

Figure 6.6 illustrates the problem encountered when multiple node pairs search for replacement paths simultaneously. If only one of the nodes from a relation affected by a failure searches for replacement paths, referred to as ordinary flooding in Figure 6.6 (a), it is possible that some replacement paths will traverse through the nodes of other affected relations. In Figure 6.6 the magenta restoration path for relation 2-5 traverses node 1, which is the source for relation 1-8. If 2-5 restores one unit of lost capacity by using the magenta restoration path, then 1-8 can restore at most two units of demand.

To defer traversing nodes of low degree, and more importantly, avoid early traversal of the source or destination of relations to be restored, both nodes from a relation affected by a failure could search for replacement paths concurrently. When this is done, the spare links incident with the source and destination of a failed working path are seized so that they may be efficiently used to restore the demand originating at that node. If ordinary flooding in Figure 6.6 is replaced with bidirectional flooding, restoration statelets would be transmitted on each of the links terminated at nodes 1, 2, 5, and 8. This would prevent any restoration path in Figure 6.6 from traversing either the source or destination of a relation to be restored.

OPRA therefore uses a double-ended or bidirectional flooding scheme. This is in contrast to the SHN described in [25], and to concurrent Capacity Scavenging which employs ordinary flooding as depicted in Figure 6.6. While two distributed span restoration algorithms published to date have employed bidirectional flooding in principle [10, 21], they used it to reduce the restoration time rather than optimize the restoration pathset. Though bidirectional flooding may decrease the restoration time of the distributed path restoration algorithm presented here, it is primarily necessary as part of the strategy to optimize the restoration pathset because it is an effective way to avoid the end-node bottleneck traversal problem. Spans local to an end node of a demand pair will by this principle be quickly seized into anchoring paths for that demand pair.


A single spare link
Restoration Statelet
0
1
2
3
4
5
6
7
8
9
X
Failure of span 8-9 affects relations 2-5 and 1-8

(a) Network topology showing one spare link per span and the initiation of ordinary flooding.

(b) Network state after restoration statelets have propagated one hop.

(c) Network state after restoration statelets have propagated two hops.

(d) Final steady state of the network until either node 1 or 5 acknowledge a restoration statelet.

Figure 6.6. Principle of End-Node Bottlenecking

## 6.3 Conclusion

A distributed path restoration algorithm based on the interference heuristic along with a bidirectional flooding scheme promises to achieve an optimal solution to the restoration problem within two seconds. The following chapters explain and analyse such a path DRA named OPRA. OPRA is one of the main contributions of this work

# Chapter 7. Distributed Interaction via Restoration Statelets

OPRA uses state based signalling as the framework for interaction between nodes, using restoration statelets for nodal interactions. The fields of a restoration statelet form part of the logical environment with which nodes executing OPRA interact in real time. This chapter explains distributed interaction via restoration statelets and the logical environment at a DCS within which OPRA functions. In discussing interaction via restoration statelets in this chapter, it is inevitable to delve partially into how OPRA works, the topic of the next chapter. The reader is asked to view such forward allusions to the algorithm as a form of introduction to concepts which are revisited in depth in the next chapter.

## 7.1 State Based Signalling via Restoration Statelets

Interactions between nodes in transport networks can take place over a reserved data link using messages explicitly addressed to neighbouring nodes, or via state based signalling as explained in chapter 2. State based signalling relies on a defined set of information attributes associated with a transmission link to communicate a change in an adjacent node's state vis-a-vis the connecting link. Unlike a messaging channel which transmits information through a single serial communication channel between nodes, exchanging information using state based signalling exploits the high degree of spatial parallelism inherent in a transport network by virtue of the links between nodes, and avoids protocol stacks.

OPRA relies on state-based signalling as the framework for interaction because it has benefits in simplicity and robustness, and is potentially much faster than message-based communication. State-based signalling requires that every port at a DCS has a receive and transmit register which is capable of storing the quasi-static information fields repeatedly applied to individual transport links. The receive register must have read status and the transmit register must have write status as seen by the processor. As explained in chapter 2, in a SONET network, restoration statelets may be transmitted by one or both of the unused bytes in the line overhead of a STS-1.

The nodes of a network in which OPRA is deployed interact solely through the restoration statelets (statelets) on the links between them. Restoration occurs as a network-level by-product of the isolated actions of each node, and the processor at each node acts only as a specialized statelet-processing engine during execution of the OPRA task.

The algorithm deployed at each node sees the outside world through the statelets arriving on the links at its site. Nodes influence each other indirectly by reacting to the statelets received, and changing the number and content of the statelets transmitted. While restoring a failure each node (DCS) acts as an isolated processor of statelets, sitting in and reacting to a sea of changing statelets.

## 7.2 Content and Uses of Restoration Statelet Information Fields

The fields of a statelet are not intended for general purpose communication. A node originating a statelet does not necessarily know who will receive that statelet or who will be influenced by it. Figure 7.1 shows the structure of a statelet, excluding the implementation specific fields for framing and data validation, which are always present. The subsequent discussion outlines each field of a statelet and the principle behind its use, deferring some details until later. The framing and checksum fields are omitted in the this discussion because their function is obvious. Explicit framing may not be needed in fact as this is derived from the SONET frame timing, where the line overhead bytes are identified and demultiplexed to line-level applications and processing functions.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| node ID | source | destination | index | Return Alarm bit | interference number | repeat count | reverse linking indicator | confirmation indicator | mode |

Figure 7.1. Information fields in a statelet

The function of many of the fields shown in Figure 7.1 are similar to those associated with a signature [25], which is used to facilitate state based signalling in the SHN. In the following explanation of the role of each restoration statelet's field, these similarities are identified.

*Node ID (NID)*: The contents and use of the NID are the same as that of the NID field in a signature. As discussed in [25], this field is always asserted by a node when it applies a statelet to a link. The NID field is assigned the network-wide identifier of the node originating the statelet. The NID is not used for explicit routing purposes as it may seem natural to assume; rather it is used to deduce link-to-span logical associations. Each DCS groups links arriving at its site into logical spans based on the property that all incoming statelets with the same NID have come from the same adjacent node, and therefore form a logical span.

Span entities are an important logical construct in distributed restoration algorithms because the geographical diversity of a network's topology is resolvable at the span level. Using the NID eliminates the requirement for stored tables describing the composition of spans at each node because each node in which OPRA is deployed can deduce link-to-span associations from the NID for itself. Although the spans of a network are easily apparent from a centralized point of view, each node in which OPRA is deployed sees only an unstructured array of ports and must deduce span associations, and other logical implications of the topology, for itself. Using the NID also ensures that the network topology as seen by OPRA is up-to-date, even in the event of maintenance or service rearrangement.

Under normal operating conditions where there are no failures a null statelet is placed on every working and spare link in a network. In a null statelet all fields except the NID are set to null.

*Source and Destination*: Source and destination fields function as a label identifying the demand pair to be restored. In the event of a failure, these fields are assigned non null values by the nodes terminating a severed working path. Tandem nodes repeat these fields but never overwrite or change them.

The source and destination fields are also used to establish the logical direction of a statelet based on the order of the node names. For example, if node x is entered into the source field and node y is entered into the destination field of the statelet transmitted on link z, then the statelet on link z is tracing a restoration path from node x to node y.

The source and destination fields are similar to the source and target fields of a signature in [25]. However, unlike the source and target fields of a signature, which identify the two nodes directly adjacent to a span failure, the source and destination fields identify the demand pair affected by a failure, and multiple demand pairs are usually affected by a single span failure. Furthermore, unlike the source and target fields of a signature, the source and destination fields of a restoration statelet do not serve as "labels" which keep the processing for one "fault instance" separate from all others. OPRA is a path restoration mechanism capable of restoring multiple span failures simultaneously without requiring that an instance of the DRA be initiated for each separate span failure. The source and destination fields simply allow a node receiving a statelet to determine whether or not it is the destination of that statelet, and if so, to which affected demand pair involving this node the received statelet belongs.

*Index*: The index field of a statelet is analogous to the index of a signature in [25]. It is used in the SHN and OPRA to uniquely identify a statelet family which may result in one link-disjoint path and to manage the contention which results from the several path construction efforts that occur concurrently in both OPRA and the SHN.

The index field is assigned a unique number by the source of a statelet at the start of the forward flooding phase. Indexing is not repeated at a source node for a given demand pair, but runs sequentially over all spans and all statelets initiated by a node, as shown in Figure 7.2. Only source nodes, i.e. the end-nodes of a failed working path, generate index numbers. Similar to the index field of a signature in the SHN, the index field of a statelet is generated once and never altered by any other node.

In Figure 7.2, if demand pairs A-B, and A-Z lost 1 and 2 units of demand, respectively, the outgoing statelets shown would correspond to the target broadcast pattern for node A when functioning as a source. The target broadcast patterns for source and tandem nodes are presented in greater detail in the following chapter.

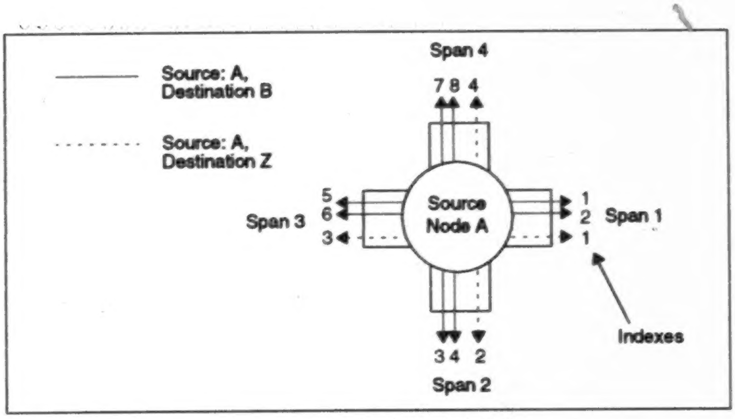


Figure 7.2. Indexing statelets

Whereas only a single Sender node in the SHN generates index numbers, all the demand pairs affected by a span failure generate index numbers in OPRA, and Index numbers are only unique for a single demand pair. Therefore, the index field alone cannot represent a sub-family of statelets like it can represent a sub-family of signatures in the SHN [25]. As shown in Figure 7.2, index 1 is used twice; once by demand pair A-B, represented in red, and once by demand pair A-Z, represented in blue. The source and destination in conjunction with the index field uniquely identify a statelet family in a network executing OPRA.

Each source, destination, and index field triplet originating at a source node becomes common to a larger set of statelets rebroadcast by tandem nodes, defining all statelets in one of the multiple concurrent flooding processes initiated by OPRA. The tandem node rules for contention amongst these flooding instances for access to available links treats statelets belonging to the same family in certain ways, and in different ways across families. The notion of competition between statelet families is central to OPRA because of its connection with the interference heuristic as discussed in the following chapter.

*Return Alarm Bit (RA bit)*: Whenever a node observes an alarm on a working link it sets the return alarm bit in the reverse direction statelet. In the case of a unidirectional failure, such as a one-way single-regenerator, splice, fiber, or connector failure, the RA bit ensures that both the source and destination of a failed working path correctly activate

OPRA. When a failure is bidirectional, the RA bit is redundant as other alarms activate OPRA in both the source and destination. All operations on the RA bit are delegated to the DCS port hardware, releasing OPRA from the task. This attribute is inherited directly from the SHN signature and simply extended here to use on an end-to-end path basis.

*Interference Number (IntNo)*: The interference number of every statelet transmitted by a node is increased by the current value of the interference number of the span used to transport the statelet. The interference number of a span is calculated by counting the number of statelets competing, not qualifying, to be forwarded on a span, and subtracting from this sum the number of paths the span can support, as shown in Figure 6.3.

Statelets initiated by a source node compete with statelets from other demand pairs for which this node functions as a tandem node for outgoing spare links. The interference number of all statelets initiated by a source are assigned the same interference number. The initial value of a statelet's interference number is determined by the value of a node's Initial Interference Number (IIN).

While the initial interference numbers of all statelets initiated by a source node are usually set to zero, they may be increased after a lost working path is restored in order to facilitate restoring at least one working path from another demand pair affected by the same failure. The target broadcast pattern of the statelet with the lowest interference number is always satisfied first to the greatest extent possible at a tandem node, so increasing the interference number of all statelets emitted by a source gives statelets from other relations a better chance of being rebroadcast at tandem nodes.

Typically, the interference number of a statelet accumulates as it traverses the network. However, the interference number of a statelet is dynamic and may decrease as well as increase as statelets on other index, source, and destination families arrive and disappear at tandem nodes. While the interference number of a statelet may decrease, it is never less than zero because the interference number of a span is never allowed to be negative. Furthermore, any increase or decrease in a statelets interference number will cause a chain reaction in which the change is propagated down each branch of a statelet's family's logical tree, that spatially extends in all directions, and forms that family's broadcast mesh.

The interference number of a statelet is the primary factor influencing the logic of tandem nodes when statelets are in contention for rebroadcast. As mentioned previously, the statelet with the lowest interference number is always rebroadcast first at a tandem node.

Repeat Count: The repeat count field is a logical hop counter with the same basic role as it had in the SHN [25]. It is part of a mechanism used to control the distance a statelet propagates in a restoration event. Associated with the use of the repeat field are two constants, the repeat-limit (also called *maxRepeats*) and the Initial Repeat Value (IRV), which are embedded in OPRA. When a source initiates a statelet, it assigns the IRV to the repeat field. *maxRepeats* is a network wide constant but the IRV is a node-specific constant so that, if desired, the range of a statelet can depend on which node acts as its source. This permits a region-specific range for restoration. The maximum number of hops that any statelet will be allowed to propagate away from the source is determined by subtracting the IRV from *maxRepeats*. The IRV can be made positive to reduce range, or negative to increase range.

While the range of a statelet is limited to *maxRepeats*, the maximum logical hop length of a restoration path found by OPRA is twice the value of *maxRepeats* because now both of the nodes from a demand pair to be restored behave as sources, analogous to Senders in the SHN [25], following a failure. Each node traversed by the logical tree from a statelet family mesh increments the repeat field and rebroadcasts the statelet, except if the repeat limit would be exceeded. Any statelet that arrives at a node with a repeat value equal to or greater than the repeat-limit is ignored.

Reverse linking Indicator (complement field): As mentioned previously, OPRA employs bidirectional flooding when synthesizing restoration pathsets. After a failure, both end-nodes terminating a severed working path, named A and Z here, begin transmitting statelets that form the base or root of a broadcast mesh from a single source-destination-index statelet family, as shown in Figure 7.3. At this stage of restoration, nodes A and Z behave analogous to Senders in the SHN which initiate an index specific, rather than a source, destination, and index specific, flooding pattern. Unless a restoration path is one hop long, the statelet's initiated by A and Z will in general be rebroadcast through the network and meet at some tandem nodes, as shown in Figure 7.3. When a statelet initiated by node A collides with a statelet initiated by node Z, the source-destination field is recognized as a match and a potential restoration path is identified. This event is called a *match* in OPRA. After a match, the reverse linking indicator of the forward flooding statelet initiated by node A is set to the value of the index of the statelet initiated by node Z, and conversely for the reverse linking indicator of the statelet initiated by node Z.

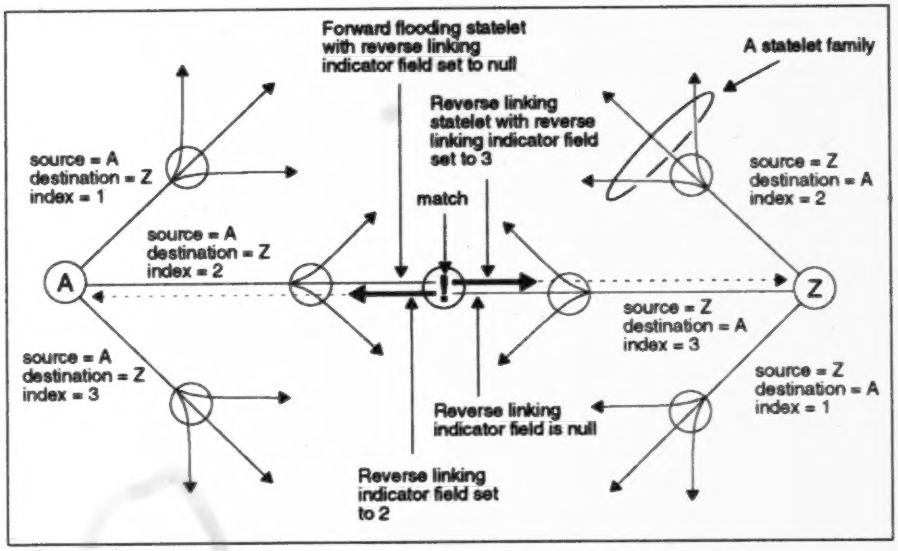

Forward flooding statelet with reverse linking indicator field set to null
A statelet family
Reverse linking statelet with reverse linking indicator field set to 3
source = A destination = Z index = 1
source = Z destination = A index = 2
match
source = A destination = Z index = 2
A
!
Z
source = Z destination = A index = 3
source = A destination = Z index = 3
source = Z destination = A index = 1
Reverse linking indicator field is null
Reverse linking indicator field set to 2

Figure 7.3. Setting the Reverse Linking Indicator in a statelet

Once the reverse linking indicator is set to a non-null value, the statelet from node A follows a path paralleling the forward flooding path of the statelet initiated by node Z, as shown in Figure 7.3. A statelet whose reverse linking indicator is set is always paired with another statelet whose logical direction is its exact complement, so the source of the statelet initiated by node A is the destination of the statelet initiated by node Z, and vice-versa. Consequently, a statelet with the reverse linking indicator set is often referred to as a *complemented* statelet, and the reverse linking indicator is often referred to as the *complement* field. Reverse linking is complete when a complemented statelet reaches the source of the forward flooding statelet with which it is matched.

When the forward flooding statelet paired with a complemented statelet does not disappear at a tandem node, the reverse linking process is similar to the one described in the SHN [25]. In this case the arrival of a complement statelet at a tandem node causes the complement condition to be propagated toward the source of the forward flooding statelet with which the reverse linking statelet is paired, and the cancellation of all other statelets on that source-destination-index family, reopening the local tandem node competition for new statelets on other families. However, unlike the reverse linking process in the SHN, a crosspoint order may not be given at this stage because the forward flooding statelet paired with a reverse linking statelet may disappear as explained in detail in chapter 8. In OPRA the forward flooding statelet paired with a complemented statelet may disappear even after reverse linking is continued locally at one node. This trait is one key characteristic which distinguishes reverse linking in OPRA from reverse linking in the SHN, and is required to synthesize a near optimal restoration pathset as tested and confirmed in the results presented at the end of this thesis.

*Confirmation Indicator*: OPRA allows a statelet to be overwritten by a statelet from another family up until reverse linking is complete. In the event that the forward flooding statelet which is paired with a complemented statelet is overwritten locally at one node, reverse linking is stopped at that node. To determine whether or not the two complemented statelets initiated after a match reach their destination, the demand pair to be restored (A and Z in Figure 7.4) perform a loop-back test using the confirmation indicator.

This confirmation phase is not an optional choice in deployment or implementation of OPRA because without it, in the event that the red reverse linking statelet reaches node A and the blue reverse linking statelet *never* reaches node Z, node A in Figure 7.4 would erroneously assume a complete restoration path existed between itself and node Z. Unlike the SHN, the state-based nature of the event-sequence during reverse linking that forms the paths of a statelet family, which in the SHN is defined by a signature's index and in OPRA is defined by a statelet's source-destination-index label, does not constitute an inherent end-to-end continuity test. Only the confirmation phase of the restoration process constitutes an inherent end-to-end continuity test in OPRA.

The confirmation phase is initiated after the complemented statelet destined for the node with the smaller ID, node A in Figure 7.4, reaches its destination. This continuity test mimics the loop-back test often performed to check a new circuit in a telecommunications network, and involves transmitting a statelet with the Confirmation Indicator bit set along the path traced by the complemented statelet. A statelet with the confirmation indicator set will be referred to as a *confirmation* or *confirmed* statelet. A confirmation statelet starts at the source with the smaller node (node A in Figure 7.4), proceeds to the destination along the path traced by the reverse linking statelet, and returns to the source if a valid restoration path exists.

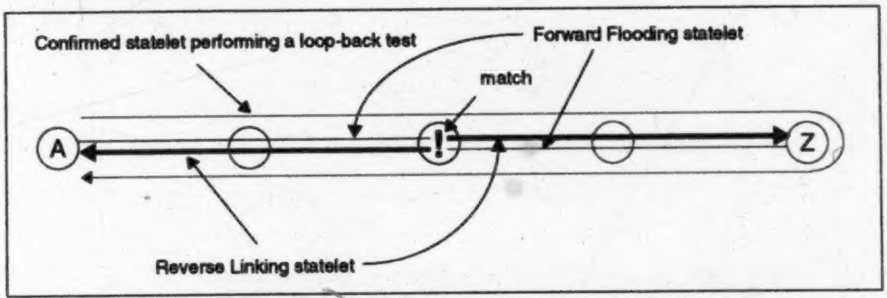

Confirmed statelet performing a loop-back test
Forward Flooding statelet
match
A
!
Z
Reverse Linking statelet

Figure 7.4. Initiating a loop-back test using the Confirmation Indicator

*mode*: The mode field of a statelet is used to identify the particular application which should be executed using OPRA's real-time distributed autonomous path-finding capability. OPRA has been primarily oriented around path restoration. This involves special considerations such as alarm detection and checking that links are in spare status before using them for rerouting. These and other aspects specific to the restoration application appear in the description of OPRA which follows.

However, the central result of the algorithm is a real-time distributed autonomous mechanism for finding multiple link-disjoint paths in a multi-graph. This basic core of the algorithm can easily be extracted from the description here. This path finding capability can also be adapted to a number of other applications. Some applications that have been recognized and studied, to varying degrees, include:

- real time path restoration (presented in this thesis)
- distributed preplanning for restoration [54]
- network restorability audit [6]
- network capacity audit [6]

- service provisioning on demand [6, 34]
- self-traffic engineering [6, 25, 46, 49]

The incorporation of this mode indicator in OPRA follows the previous work done in [25] where it was proposed that the SHN be extended in this way to permit such a variety of applications built upon the basic pathset synthesis utility the algorithm provides. It seems opportune therefore to explicitly provide for such generalized applications in defining OPRA's statelet's.

## 7.3 Factors Determining Statelet Length

The factors determining the bit size of the various fields of a statelet are based on several planning and implementation considerations. A statelet's length must be specified so that network growth is allowed for. Two measures of a network's size are reflected in the statelet:

(a) maximum number of nodes: this determines the length of the three node-name fields - NID, source, and destination.

(b) maximum number of working links on any one span: this determines the maximum number of bits that could be required for the index field of a statelet.

Not every field of a statelet is dependent on the size of a network. The RA field requires only a single bit to echo a receive failure to the far-end node.

The interference number of a statelet depends on the number of statelet's competing to restore lost working paths after a network failure. In the event of a single span failure, the maximum interference number of a statelet can be estimated by multiplying the link-size of the largest span in a network by the degree of the largest node in the network and by the hop length of the longest restoration path allowed. A conservative estimate of the IntNo field's size would set it large enough to accommodate an interference number of this magnitude.

When 4 bits are used to transport the repeat count, a maximum repeat count limit of 16 spans is possible, and restoration paths are restricted to a maximum of 32 hops. A 32 hop maximum restoration path length limit is certainly sufficient for an extensive search of the solution space in even the largest networks used to test OPRA's ability to synthesize a near optimal restoration pathset.

The reverse linking indicator is set to the value of a statelet's index after a match, as explained in the previous section. Therefore, the size of a statelet's complement field should be the same size as a statelet's index field.

The confirmation indicator is always assigned a unique value upon initiating a loop-back test. The size of a statelet's confirmation field should be large enough to assign a unique value to each restoration path initiated by a node. The maximum number of restoration paths for a given demand pair will never be more than the size of the largest span in a network. Therefore, an estimate of the confirmation indicator's bit size is the binary number for the link-size of the largest span in a network.

The size of a statelet's mode field depends on the number of applications which may use OPRA's path finding capability. OPRA may be adapted to a number of provisioning, network audit, and network functions. The reservation of 3 bits for the mode field would allow development of up to 8 different applications which reuse OPRA's kernel at each node.

The size of a statelet's parity or checkbits depends upon whether simple repetition rather than parity checking is used to verify a statelet's fields. Simple repetition could reduce the effective statelet transfer speed by a factor which could be significant enough to affect real time performance. A single parity check bit could therefore be used to qualify received statelets, eliminating the need to wait for repetition in most cases. However, 2 errors within a statelet would defeat a one bit parity check. Ultimately, a 5 or 6 bit Cyclic Redundancy Check (CRC) would be the most conservative allocation to make. CRC-6 computation is simple to implement in a shift register circuit, and provides virtually zero probability of an errored statelet being interpreted as a state change, especially if repetition/persistence is also relied upon to generate a changed statelet event in a DCS port.

When one or both of the unused bytes in the LOH of a SONET transport signal is used to transmit statelets, no word alignment field is required to identify the start of a statelet. However, in some methods for statelet transport [26], it is impossible to use the timing attributes of the carrier signal to frame on a statelet. In this case one constantly toggling bit will allow simple statistical frame alignment on a statelet. If the statelet word alignment bit toggles in every statelet repetition, framing is simple and fast. Fast, robust, framing can also be derived from the presence of an embedded CRC or FEC code, without explicit allocation of framing overhead [26].

Based on the information just presented, and the size of the actual transport networks shown in Figures 4.5 to 4.8, the following statelet field sizes were assumed in all of the OPRA tests performed:

| statelet Field | Number of bits allocated | |
|---|---|---|
| NID | 8 | |
| source | 8 | |
| destination | 8 | |
| index | 8 | |
| RA | 1 | |
| IntNo | 9 | |
| repeat count | 4 | |
| reverse linking indicator | 8 | |
| confirmation indicator | 8 | |
| mode | 3 | |
| CRC | 6 | (optional) |
| statelet word alignment | 1 | (need depends on transport environment) |
| Anticipated statelet bit length | 72 | |

This anticipated statelet bit length is a natural allocation for a SONET OPRA implementation because a single SONET line overhead byte yields 8 bits of information. Given 9 bytes of statelet contents, plus a byte to align on the repetition sequence, a statelet's insertion time on a span would be 1.25 msec.

## 7.4 Classifying Restoration Statelets

In the description of OPRA, statelets are classified into 3 groups depending on which information fields are assigned null values. The names given each group reflect the characteristics of the statelets associated with that class as shown in Table 7.1.

Table 7.1. Classifying Restoration Statelets

| statelet class | Common characteristics |
|---|---|
| Forward flooding statelet | NID ≠ null<br>source ≠ null<br>destination ≠ null<br>index ≠ null<br>IntNo ≠ null<br>repeat count ≠ null<br>mode ≠ null<br>**Reverse Linking Indicator = null**<br>**Confirmation Indicator = null** |
| Reverse Linking or Complemented statelet | **Only the Confirmation Indicator field from this class of statelets is null** |
| Confirmation statelet | No fields from this class of statelets are null |
| Null statelet | Each field from this class of statelets except the NID is null |

## 7.5 Relationships between Restoration Statelets

As alluded to in section 7.2, special relationships may exist between various statelets during a restoration event. These relationships are formally defined here.

*__The Precursor Relationship__*: For each statelet family present at a tandem node, the port at which the root of the rebroadcast tree for that statelet family is found is called the precursor for that statelet (see Figure 7.5). Transmitted statelets from the same family as the precursor are referred to as the branches of the rebroadcast tree. The precursor of a statelet family usually has the lowest interference number of all incoming statelets at the node on the same source-destination-index label and always is the root of the broadcast tree for that mesh. A precursor is a directed one-to-many relationship. One incoming statelet can be the precursor for many outgoing statelets, and every outgoing statelet has only one precursor. The precursor relationship strictly applies between statelets, meaning the precursor of a source-destination-index label may change ports at a node while restoring a failure.

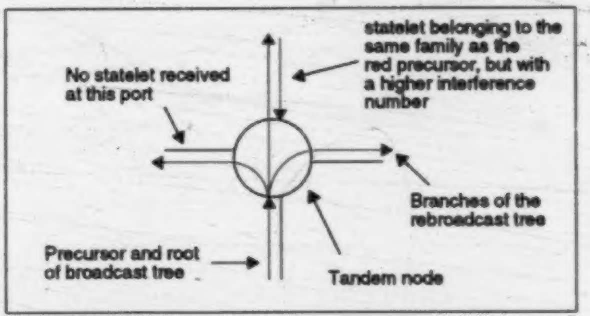

statelet belonging to the same family as the red precursor, but with a higher interference number
No statelet received at this port
Branches of the rebroadcast tree
Precursor and root of broadcast tree
Tandem node

Figure 7.5. Precursor Relationships at Tandem Nodes

*Restoration Statelet Families*: A family is the network-wide set of statelets with the same source, destination, and index. All of the statelets from the same family have the same logical direction, i.e. from source to destination regardless if the statelet is classified as forward flooding or a complemented statelet. Each statelet family may lead to the establishment of at most one restoration path. During forward flooding, a family of statelets expands outward from the source, tracing a path which is always connected back to the source via the sequence of precursor relationships established through preceeding tandem nodes in a statelet family's mesh.

*Matched Restoration Statelets*: Two precursors, x and y, form a match when the source of statelet x is the destination of statelet y, and the source of statelet y is destination of statelet x. Before a forward flooding statelet is rebroadcast at a tandem node, each port is checked to see if the received statelet matches any other received statelet. Matched statelets do not need to have the same index, but must both be precursors and be on the same source-destination label. Only statelets whose confirmation and reverse linking indicator fields are null can establish a match, i.e. a match can only be defined between forward flooding statelets. While a match condition arises frequently in OPRA, reverse linking is not always initiated. Initiating reverse linking after a match in OPRA is carefully controlled to minimize initiating reverse linking procedures which are destined to fail, and maximizing those which lead to the creation of a near optimal restoration pathset as explained in detail in the following chapter.

## 7.6 The Logical Environment of a Node

It is necessary to define a hardware environment in which OPRA executes in order to establish a model logical programmers can reference when establishing and specifying OPRA. The hardware environment described in this section is not the only implementation of OPRA possible, but one which is consistent with the hardware environment developed to support the SHN [25].

OPRA views the ports which interface the links of the transport network as the logical environment of a node. The port is the location at which statelets are received, and through which statelets are applied to a link. Each DCS port has storage registers for one transmit statelet and one receive statelet. In addition to the statelet registers, a register is required to store port specific, rather than link specific, state information. The Port Status Register (PSReg) stores port specific information pertaining to both the transmit and receive links terminated at that port. Section 7.2 covered the details of transmit and receive port registers; this section covers details of the PSReg.

### 7.6.1 Port Status Register

The PSReg includes the following one-bit status fields: Line Alarm, Path Alarm, spare, activated, Restoration Statelet Change Interrupt Enable (RSCIE), Alarm Interrupt Enable (AIE), Alarm Indication Signal (AIS), and Receive Restoration Statelet (RxRS). It also includes an associated port (assoc-port), an associated span (assoc-span), and Trace field as shown in Figure 7.6.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Line Alarm | Path Alarm | AIS | spare | RSCIE | AIE | activated | RxRS | assoc-port | assoc-span | Trace |

Figure 7.6. Information fields in the PSReg

The function of many of the fields shown in Figure 7.6 are similar to those associated with the port status register presented in [25] for the SHN. In the following explanation of each field these similarities are identified.

*Line Alarm*: A Line Alarm occurs when a port terminates a link that is cut, as shown in Figure 7.7. Any number of incoming link conditions such as signal loss, framing loss, clock recovery, loss of lock, high BER, etc. may trigger a line alarm signal at the LTE as described in section 2.1.1.

*Path Alarm*: A Path Alarm occurs when a port terminates a failed working path. Any number of incoming link conditions such as signal loss, framing loss, clock recovery, loss of lock, high BER, etc. may trigger a path alarm signal at the PTE as described in section 2.1.1. The path alarm informs a node it will be involved as a source and destination in the restoration process.

*Alarm Indication Signal (AIS)*: The AIS bit is used to indicate which ports lost live traffic due to the failure of either a working or previously found restoration path. The AIS bit of all ports in a failed working path or a failed restoration path are set to true. OPRA uses the AIS field, along with the Line Alarm and spare bits, to determine which ports may be released after a failure when performing stub release. Only ports with AIS = true, spare = false, and Line Alarm = false are eligible for stub release, as shown in Figure 7.7. When a working port is added to the pool of spare capacity available for use in restoration, the activated and AIS fields of the PSReg are set to false.

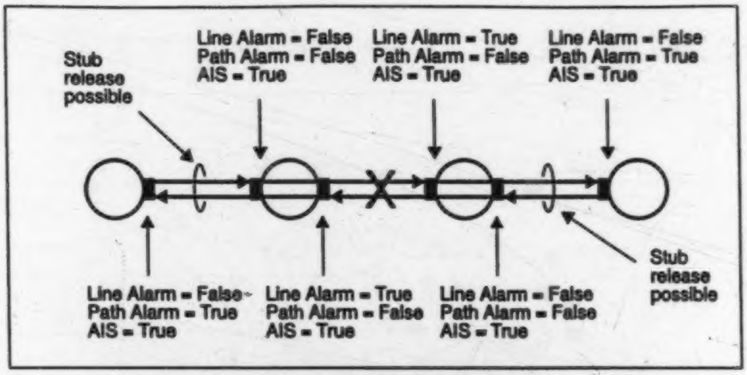

Stub release possible
Line Alarm = False
Path Alarm = False
AIS = True
Line Alarm = True
Path Alarm = False
AIS = True
Line Alarm = False
Path Alarm = True
AIS = True
X
Line Alarm = False
Path Alarm = True
AIS = True
Line Alarm = True
Path Alarm = False
AIS = True
Line Alarm = False
Path Alarm = False
AIS = True
Stub release possible

Figure 7.7. Alarm detection related PSReg contents after the failure of a working path

*spare*: The spare bit is set by the DCS in the course of normal provisioning activities and never changed by OPRA. Under normal network conditions of no network failures, the spare bit of those ports not carrying any demand is set to true, and the spare bit of those ports carrying demand is set to false. Unlike the use of a port's spare bit in the SHN [25], the spare bit here is not used by OPRA to identify those ports available for use in restoration at the time of a failure because under abnormal network conditions, such as after restoring a failure, a port with its spare bit set true may be part of a restoration path used to restore an earlier failure. OPRA uses the spare bit along with the Line Alarm and AIS bits to determine which ports can be released after a failure when performing stub release, as explained previously.

*activated*: The *activated* bit indicates whether a port is logically connected, and identifies those ports available for use in restoration at the time of a failure. If the activated field of a port is false, OPRA may use that port to transmit a statelet. Under normal network conditions the *activated* bit of a spare port is set false, and true for a working port. After stub release, the *activated* bit of a working port added to the pool of spare capacity available for use in restoration is set to false.

The *activated* bits of ports part of a forward flooding process are always set to false. The *activated* bits of ports part of a reverse linking process are always set to true.

(As described in section 7.2, a pair of logically connected ports at a node part of a reverse linking process whose *activated* bits are set to true are not physically cross-connected in a DCS's switching matrix until OPRA confirms a path's continuity (see Figure 7.4).) Unlike the spare bit, OPRA toggles the value of the *activated* bit between true and false when a link becomes part of a reverse linking and forward flooding process respectively. During the course of restoration, the activated field of a port may toggle between true and false many times while the paths traced by all statelets coalesce into a near optimal replacement pathset.

The *activated* field is not simply an indicator if a link is in statelet use or not, rather it indicates when a link is part of a reverse linking process, which may or may not lead to the successful synthesis of a restoration path. During the synthesis of a restoration path a single link may be part of multiple distinct reverse linking and forward flooding processes as described in detail in the following chapter. Before, during, and after, a restoration event, the *activated* field indicates which ports are not part of a reverse linking process and OPRA can use to transmit new forward flooding statelets.

*Restoration Statelet Change Interrupt Enable (RSCIE)*: The receipt of alarms and statelets generates interrupts sent to OPRA for processing. A port's interrupts can be masked by setting the RSCIE false. Setting the RSCIE false prevents a port from participating in a restoration event, effectively turning that port off. The use of the RSCIE is the same as the SCIE in the SHN [25].

*Alarm Interrupt Enable (AIE)*: AIE is used to inhibit alarms on failed links. If the AIE is set false, a port is prevented from receiving alarms. Usually the AIE is only set false after a link receives an alarm, ensuring that repeated alarms are prevented from interrupting OPRA. The use of the AIE is the same as the AIE in the SHN [25].

*Receive Restoration Statelet (RxRS)*: Whenever a statelet is received and confirmed, possibly by a hardware level check for CRC or repetition, and that statelet is different from the current contents of the receive register, the RxRS bit is set to true. This informs OPRA that a statelet-change event has occurred in that port. It is possible that multiple restoration statelets and/or alarms arrive at a node while OPRA is responding to an interrupt. Only after OPRA has completed processing an interrupt does it acknowledge newly received statelets as indicated by a port's RxRS field. The use of the RxRS is the same as the RSDEL in the SHN described in [25].

assoc-port and assoc-span: *assoc-port* and *assoc-span* store the identity of the port to which this port is connected logically by the precursor relationship during restoration, or physically connected via the local DCS matrix after restoration. During restoration, the *assoc-port* and *assoc-span* fields of all ports point to the location of the precursor for the outgoing statelet transmitted on that port. Given a completed restoration path, the *assoc-port* and *assoc-span* fields record physical cross-connect information.

Given that a port's span can always be obtained by reading the NID, only the *assoc-port* field is fundamentally required for the OPRA implementation described in this thesis. The *assoc-span* field is for convenience only. Furthermore, if the ports on a DCS have a node-global numbering assignment, the *assoc-span* field is not even needed for convenience, because a port in this case can be uniquely identified at a node by its port number alone, without specifying the span in which that port lies.

*Trace*: For each node terminating a working path, the *Trace* field records the ID of the far-end node. The *Trace* field is required to identify the demand pair to be restored after a failure in a path restorable network. While the path overhead of a SONET STS signal could carry the node ID of a signal's source, possibly using the J1 trace byte, such an application has not been standardized in SONET. Consequently, OPRA stores the ID of the far-end node of a working path terminated at a port at the time a working path is provisioned in the *Trace* field.

## 7.7 Summary

The receive statelet register, transmit statelet register, and Port Status Register collectively form the memory space in which all data needed by OPRA are present and automatically updated by hardware in response to external events outside the node itself. All statelet processing rules are defined in terms of operations on the set of all registers at the active ports of a given DCS node. From the viewpoint of OPRA, the host environment is therefore an array of logical records called ports.

Together, the RxReg, TxReg, and PSReg form the logical environment within which OPRA operates. OPRA perceives its environment through these registers only. All of the functions OPRA performs while restoring a failure require manipulating one or more fields of these registers. The following chapter begins a detailed explanation of these functions.

# Chapter 8. Description of the Optimized Distributed Path Restoration Algorithm (OPRA)

This chapter presents a detailed description of OPRA emphasizing the node logic to analyse and react to the various events encountered during the restoration process, and thereby determine the protocol response by a node to those events. The overall action of OPRA will be introduced in seven conceptual stages:

1) activation,
2) bidirectional selective forward flooding,
3) recognizing a match,
4) reverse linking,
5) restoration path confirmation,
6) traffic substitution, and
7) terminating a restoration event.

In reality, these phases are not exclusive, separate, or sequential, in time or space. They occur in a heavily concurrent manner while restoring a failure. However, OPRA is described in the following sections as if these phases were separate and non-overlapping in time and space, solely to convey a functional understanding of the algorithm.

## 8.1 Finite State Machine (FSM) Representation of OPRA

At its highest level of abstraction, OPRA is implemented as a Finite State Machine (FSM). As with the SHN, all processing by OPRA is of an event-driven nature using FSM techniques to encode all behavioural rules, and can be described in terms of two primary event types that drive the FSM: a statelet arrival and "alarm" events. Restoration statelets as well as line and path level alarms generate interrupts which initiate transitions in the algorithm starting from the idle or normal state (state 14 in Figure 8.1), traversing one or more action blocks (states 1 - 13 in Figure 8.1), and ultimately returning to the idle state.

Action blocks are referred to as forced states in OPNET [52], and the term forced state is used throughout this thesis. OPNET is a comprehensive development environment supporting the modelling and performance evaluation of distributed systems by discrete event simulation. OPNET was used to implement and test OPRA, and the C-code and OPNET kernel procedures needed to model OPRA in OPNET are given in [40].

Each of the seven conceptual stages presented at the start of this chapter require traversing one or more of the action blocks shown in Figure 8.1. States are mutually exclusive and complementary, meaning that OPRA is always exactly in one FSM state or action block: more than one state or action block may never be occupied at a time. OPRA moves between states in response to the interrupts it receives. The transitions shown in Figure 8.1 that depart from the idle state indicate which action block is executed, and the event condition that each change requires.

The condition which must be evaluated to determine whether or not OPRA should enter the transition's destination state appears as a parenthesized label next to the arc. Transitions that have non-empty condition expressions are depicted as dashed arcs; unconditional transitions are depicted as solid arcs. All transitions that depart from a state are evaluated before OPRA progresses to any transition destination. Transitions that emanate from a state are mutually exclusive at the time they are evaluated (no two can simultaneously be true), and complementary (at least one is necessarily true).

The light state (state 14) shown in Figure 8.1 is the only state in which OPRA may pause between interrupts, and is consequently referred to as an *Unforced State* in OPNET terminology. Unlike state 14, OPRA does not wait for an interrupt while in states 1 - 13. As mentioned previously states 1 - 13 represent action blocks, and are referred to as forced states in OPNET terminology. While forced states cannot represent actual system states, they are useful to graphically separate the actions and control flow decisions OPRA must perform in response to an interrupt. Graphically separating definitions of decisions and actions into action blocks or unforced states as shown in Figure 8.1 provides better modularity for specification, as well as a more visually informative FSM.

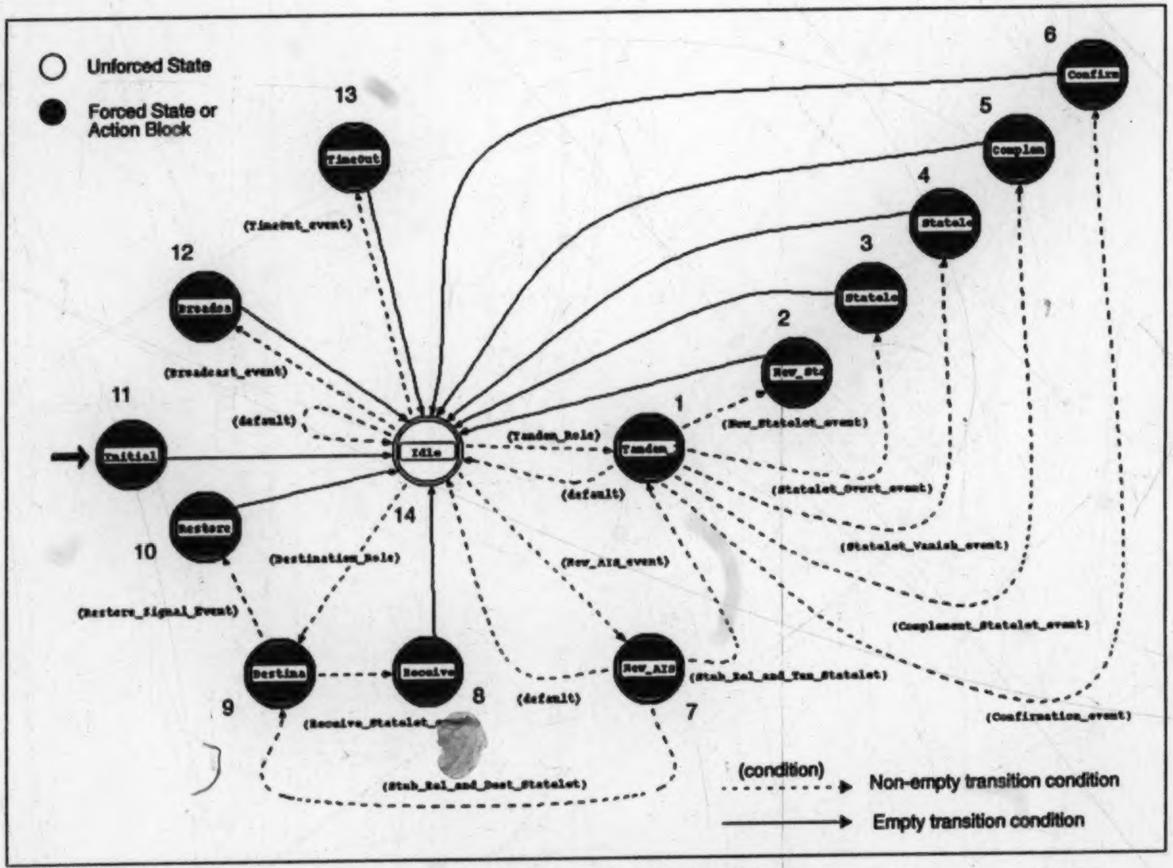

Unforced State
Forced State or Action Block
13
TimeOut
(TimeOut_event)
12
Breadca
(Broadcast_event)
11
Initial
(default)
Idle
10
Restore
14
(Destination_Role)
(Restore_Signal_Event)
9
Destina
8
Receive
(Receive_Statelet_event)
(Stub_Rel_and_Dest_Statelet)
(default)
7
New_AIS
(New_AIS_event)
(Stub_Rel_and_Tan_Statelet)
(Tandem_Role)
(default)
1
Tandem_
(New_Statelet_event)
2
New_Sta
3
Statelc
(Statelet_Overt_event)
4
Statelc
(Statelet_Vanish_event)
5
Complem
(Complement_Statelet_event)
6
Confirm
(Confirmation_event)
(condition)
Non-empty transition condition
Empty transition condition

Figure 8.1. OPRA State Transition Diagram

## 8.2 Initializing and Activating OPRA

After initializing the PSReg, RxReg, and TxReg, as well as a few memory locations OPRA uses, activation occurs when a path-level alarm is received at a node. In reality the individual port alarm detections may be distributed in time, and OPRA is tested later under both staggered alarm and concurrent alarm conditions. Detection of an alarm raises an interrupt in response to which OPRA identifies all working ports that have an alarm and all logical spans and links present at a node. The functions associated with the activation of OPRA are categorized and implemented using three states shown in Figure 8.1: state 11 - *Initialization*, state 14 - *Idle*, and state 7 - *New_AIS*. The following description of the activation of OPRA details the contents of these three states, i.e. the actions OPRA must perform whenever it transits states 11, 14, or 7.

### 8.2.1 Initialization (State 11, *Initialize*)

At the heart of OPRA is a set of predefined responses to various events encountered while restoring a network failure. These responses are specified using the C programming language and a few OPNET kernel procedures. Due to the unique nature of OPRA, there were no OPNET kernel procedures which contained or encoded any of the logic necessary to correctly process a statelet or alarm. A few OPNET kernel procedures are used to manipulate linked lists. These linked lists, as well as other memory structures used in the C code specification of OPRA, are initialized in state 11 when OPRA is loaded into a DCS.

During initialization, the degree and ID of a node, as well as the total number of links per span is determined. Then OPRA initializes four lists to increase its speed of execution by avoiding frequent searches of the whole node to find ports satisfying various statelet conditions. The first list, *rec_set*, is used to track all non-null statelets received by a node. The second list, *spares*, identifies all links available for use in restoration. The third list, *dmd_pair_list*, is used to record the amount of demand lost by this node and the IDs of the end nodes terminating the failed working paths sourced by this node. The fourth list, *precursors*, is used to track the port and span of a statelet family's precursor. A copy of the first two lists, *rec_set* and *spares*, is initialized for each span terminated at a node, while only a single copy of the third and fourth lists, *dmd_pair_list* and *precursors*, is initialized at a node.

The procedures described above executed upon transiting state 11 are summarized below:

1. Initialize variables.
2. Determine this node's degree.
2. Create one *rec_set* and *spares* list per span
4. Create one *dmd_pair_list* and *precursors* list per node.

### 8.2.2 Idle State (State 14)

After initialization, OPRA enters the idle state. This state is the only state in which OPRA can pause between interrupts, and the only static state of the state transition diagram shown in Figure 8.1. Consequently, all transitions in Figure 8.1 begin and end here. Which trajectory should be followed in response to an interrupt is determined by the conditions or logic associated with each transition.

Upon entering the idle state, the transmit statelet registers which OPRA changed while processing an interrupt are latched onto the outgoing links of each altered port. Latching a transmit register onto a link updates that link's outgoing statelet. Registers are latched onto links in the same order in which they were updated internally in OPRA's execution sequence.

OPRA will remain in the idle state until the arrival of a statelet or until an alarm occurs. Before leaving the idle state, OPRA determines which span and port originated the interrupt.

When leaving the idle state, if a new restoration task is being started, OPRA builds a list of all ports available for use in restoration. This list is named *spares* and is built after the start of a restoration event rather than maintained independently at a node so that OPRA can rely on the integrity of the data depicting the actual network structure at the time of a failure.

The procedures executed upon transiting state 14 are summarized below:

1. Update the outgoing links at a node if necessary.
2. Determine which port issued the interrupt.
3. Populate the list *spares* with all ports available for restoration when starting a new restoration event.

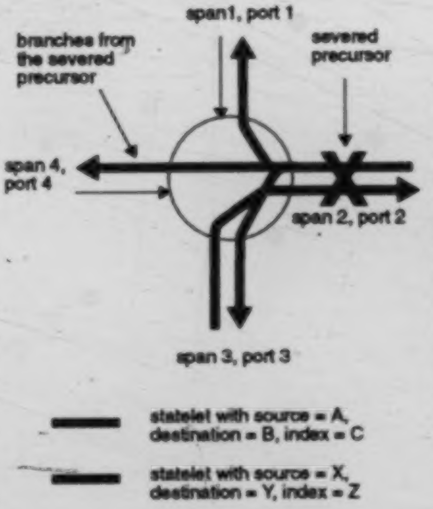
The transition which should be followed when exiting the idle state is determined by evaluating the conditions shown in parentheses in Figure 8.1. A restoration event is started when some node in the network detects a path level alarm. Detection of a path level alarm raises an interrupt causing OPRA to determine whether the AIS is true for the port initiating that interrupt. If so, the transition *New_AIS_event* is followed from state 14 to state 7.

### 8.2.3 Receipt of an Alarm (State 7, *New_AIS*)

As shown in Figure 7.7, whenever a path or line level alarm is detected at a node, AIS is true. Furthermore, AIS is true for every port in a failed working path even when no line or path level alarms exist. While a port showing only a AIS alarm cannot initiate a restoration event, that port can be added to the pool of spare capacity available for use in restoration. This is referred to as stub release. Since state 7 effects the local node's role in stub release as well as alarm processing, the transition condition from state 14 to state 7 is evaluated using the PSReg's AIS field, rather than the line or path alarm fields.

The first function performed transiting state *New_AIS* determines if a path or line alarm exists on the port initiating the interrupt. If a path alarm exists, the ID of the far end node terminating this failed working path, stored in the *Trace* field, is entered into a list which records the number of alarms received per affected relation, *dmd_pair_list*. In addition, state *New_AIS* releases the surviving portions of a cut working path when stub release is employed by setting the AIS and *activated* fields of the PSReg to false, and commanding the release of crosspoints between the ports on the failed working path.

If a port supporting a restoration path, rather than a working path fails, the source, destination, and index of the restoration path must be determined. This information can easily be determined from the transmit statelet register of the failed port. However, in the event that a precursor seeking a restoration path, rather than a completed restoration path, is severed, as shown in Figure 8.2, the information on the transmit statelet register of the failed port will not identify the source, destination and index of the failed precursor. Furthermore, after the precursor is severed, all of the fields in the receive statelet register on the failed port (port 2 in Figure 8.2) are set to null.


span1, port 1
branches from the severed precursor
severed precursor
span 4, port 4
span 2, port 2
span 3, port 3
statelet with source = A, destination = B, index = C
statelet with source = X, destination = Y, index = Z

Register Fields *immediately* after the failure of span 2

| statelet |  | source | dest. | index | assoc-span | assoc-port |
|---|---|---|---|---|---|---|
| Span/Link | In/Out |  |  |  |  |  |
| 1 | In | null | null | null | 2 | 2 |
|  | Out | A | B | C |  |  |
| 2 (before failure) | In | A | B | C | 3 | 3 |
|  | Out | X | Y | Z |  |  |
| 2 (after failure) | In | null | null | null | 3 | 3 |
|  | Out | X | Y | Z |  |  |
| 3 | In | X | Y | Z | 2 | 2 |
|  | Out | A | B | C |  |  |
| 4 | In | null | null | null | 2 | 2 |
|  | Out | A | B | C |  |  |

Figure 8.2. Identifying a severed precursor

To identify the family of a severed precursor as shown in Figure 8.2, the *assoc_span* and *assoc_port* fields of each port must be searched until a pointer to the severed precursor is found. Whether a restoration path or a precursor existed on the severed port, the *assoc_span* and *assoc_port* fields identify the port to which this port was connected, and the family of the statelet received on the severed link. For example, in Figure 8.2, the *assoc_span* and *assoc_port* fields of span 1, port 1 point to the precursor on span 2, port 2, identifying span 1, port 1, as a branch of the severed precursor. When the branch of a severed precursor is identified, the family of the precursor can be identified because a precursor and its branches all belong to the same statelet family.

Once the family of the statelet received on the severed port is identified, all outgoing links transmitting a statelet belonging to the same family are either cancelled, or possibly re-rooted. Re-rooting a precursor is only performed when a new precursor for the statelet family on the severed port is found, and requires updating the *assoc_span* and *assoc_port* fields of all the severed precursor's branches to point to the new precursor. If a replacement precursor cannot be found, or if a restoration path carrying live traffic was severed, all of the links transmitting statelets belonging to the same family as the statelet received on the severed port are cancelled. That port is then removed from the list of ports available for use in restoration and from the list of ports receiving statelets.

OPRA was designed to contend with the failure of working paths as well as completed restoration paths so that:

1. OPRA is capable of restoring restoration paths synthesized in a previous restoration event in the same network, and
2. OPRA is free to establish restoration paths using any surviving spare links on a partially cut span, as explained below.

Given that it is impossible in reality to sever all links in a span simultaneously, partial and complete span cuts are indistinguishable from OPRA's node-local point of view at the start of a restoration event. Therefore, OPRA may establish restoration paths over a span which are destined to fail when the remaining spare links on that span are severed. Without apriori knowledge whether a partial or complete span failure has occurred, and to exploit any surviving spares on a partially cut span, OPRA needs to be able to contend with the failure of both restoration paths and working paths.

Alternatively OPRA can avoid using any span containing at least one link with an active Line Alarm signal, sacrificing the ability to use any surviving spares on a partially cut span, and thereby avoid dealing with the failure of completed restoration paths synthesized in the same restoration event. Assuming that once one link on a span fails, all links on that span will eventually fail, simplifies a DRA's design and allows span DRAs like the SHN to begin synthesizing restoration paths for all working links on a span containing at least one failed working link. This is referred to as preemptive activation in [33] and is practical in span DRAs like the SHN, but not necessarily in path DRAs like OPRA, because the failure of one working path at a source/destination node distant from a failure does not mean all working paths from that demand pair traverse the severed span and are destined to fail.

The procedures executed upon transiting state 7 are summarized below:

1. Record the demand pair affected by the failure of a completed restoration path or a working path as identified by the alarm.
2. Release the surviving portions of a cut working path if employing stub release.
3. If an uncomplemented precursor failed, rather than a completed restoration path or a working path, reroot the branches of that precursor onto a new precursor, otherwise cancel its branches.

Which transition is followed upon leaving the *New_AIS* state depends on whether the port which issued the interrupt received an alarm, or an AIS signal on a working link destined for stub release along with a forward flooding statelet simultaneously. In the event an alarm was received, the default transition is followed back to the idle state. In the event a statelet was received on a port just added to the pool of spare capacity available for restoration, either state 9 or state 1 follow. State 9 follows if this node is the destination of the statelet just received, otherwise this node functions like a tandem node and state 1 follows.

## 8.3 Bidirectional Selective Forward Flooding

After the arrival of a path level alarm at a node, that node initiates forward flooding statelets on its spare links. Each primary statelet issued by a source from a given demand pair is stamped with a locally unique index value. The basic target broadcast pattern at a *source* node aims to transmit on all spans this node terminates a number of statelets equal to the number of working paths lost by this node. Through rebroadcast at tandem nodes, each primary statelet will become the root of a tree of logical associations between statelets, all of which have the same source, destination, and index value in common. The basic target broadcast pattern of a *tandem* node aims to forward one restoration statelet on one link on all spans terminated at the node except the span on which that restoration statelet arrived. Physically, the expansion is away from the source in all directions, on each statelet family. There is no preferential orientation towards the destination, because neither the source nor any other nodes know which direction is "towards" the destination.

Statelets propagate at carrier velocities and appear within milliseconds as statelet arrival events at the ports of neighbouring DCS machines. The ensuing statelet interrupt generated by the port causes the operating system of those nodes to invoke OPRA. Nodes awakened by a non-null statelet with no line or path alarms on the associated link behave as tandem nodes performing the functions associated with states 1 - 6.

The main function of a tandem node at this stage is selective rebroadcasting of statelet's based on their interference number. The network-level effect of selective rebroadcast is to activate yet more nodes as tandems. In this way, all nodes within a certain range of the source are alerted and develop an interacting mesh of rebroadcast patterns on each statelet family. The range of influence of the forward flooding phase is controlled by the repeat field of a statelet. As more tandem nodes become involved, the interference patterns of their rebroadcast efforts becomes increasingly complex and rich with information about the network's topology and capacity. The interference heuristic is responsible for mediating the competition between broadcast patterns such that a near optimal restoration pathset is found.

The basic functions associated with selective forward flooding are reused in five of the 14 states shown in Figure 8.1: state 12 - *Broadcast*, state 1 - *Tandem_Node*, state 2 - *New_Statelet*, state 3 - *Statelet_Ovwrt*, and state 4 - *Statelet_Vanish*. The contents of these five states are described next.

### 8.3.1 Forward Flood Management Logic (State 12, *Broadcast*)

The broadcast state is responsible for computing the broadcast pattern at a node. Computing the broadcast pattern at a node entails transmitting new forward flooding restoration statelets, but it does not involve cancelling statelets on other links by pruning the branches of a precursor, and/or transmitting a confirmation statelet.

The transition from the idle state to the broadcast state occurs when OPRA is issued an interrupt indicating that the broadcast pattern should be computed. The polling mechanism used to service all ports at a DCS during a restoration event, described in detail in chapter 10, is responsible for issuing this interrupt.

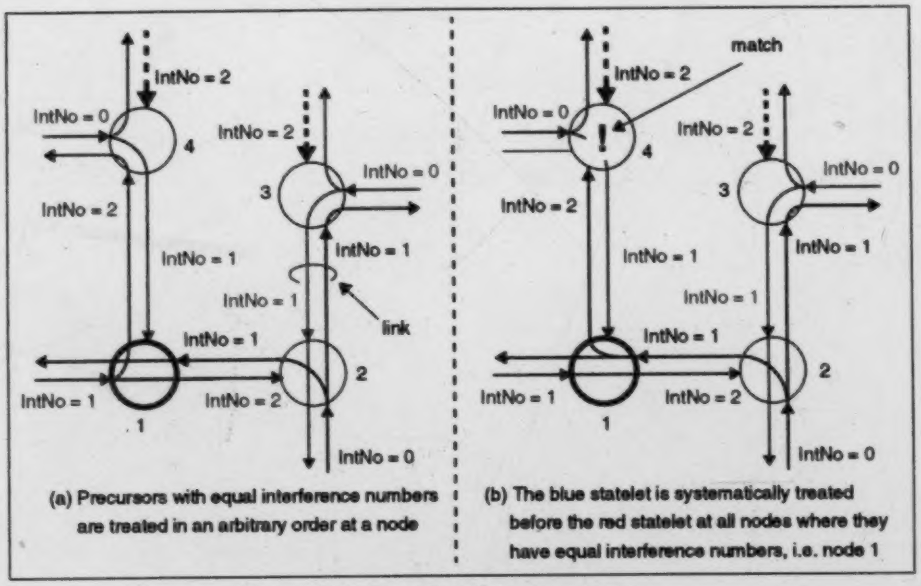
Upon entering broadcast state 12, a list of all precursors present at the node is compiled. The best statelet from a family is found and designated that family's precursor. The best precursor for a given statelet family has the lowest interference number of all forward flooding statelets belonging to the same statelet family. If two or more statelets satisfy this condition, the one with the lower repeat count is chosen to be the precursor. If two or more statelets are still eligible to be the precursor, the one which was the precursor previously is chosen, otherwise one of the remaining candidates is designated the precursor for that family arbitrarily. Precursors are only found for statelet families which have not established a restoration path traversing this node, as indicated by the *activated* field of a port.

In path restoration it is possible that a node will simultaneously source statelets as well as rebroadcast statelets from other relations as a tandem node. For the statelets initiated at a node, the source node itself acts as the initiating precursor as if an internal "failed" working link existed. To ensure a source has the opportunity to initiate its own statelets on some or all of the spare links leaving it, a flag named SOURCE-FLOOD-INDICATOR is entered into the list of precursors whenever a node has its own demands to be restored.

The list of precursors, is sorted from lowest to highest interference number. The SOURCE-FLOOD-INDICATOR is assigned an interference number equal to the IIN (Initial Interference Number). The precursor at the head of the list is given the opportunity to establish its broadcast pattern first. Subsequent entries in the list then satisfy their target broadcast pattern to the fullest extent possible, using the remaining spare surviving out going links after any stub release has occurred.

Whenever the value of the IIN is zero, source flooding precedes the broadcast of all other precursors because source flooding precedes extending the broadcast mesh of an externally arising precursor in the event a precursor's interference number is equal to the IIN. If two or more precursors have the same interference number, the one with the lower destination ID is preferred. If a tie still cannot be resolved, the statelet with the lower source ID is chosen. While the repeat count is used to resolve a tie between statelets from the same family vying to be that family's precursor in the event both have the same interference number, the repeat count is not used to resolve a tie between precursors with the same interference number when sorting the list of precursors. Using the method described above to sort the precursor list ensures one precursor will precede another in the precursor lists constructed at all nodes. Consistently satisfying one statelet family at all nodes in the event two or more precursors at a node have the same interference number facilitates finding a match and helps decrease the restoration time, as explained below.

Consistently satisfying the broadcast pattern of one statelet family at all nodes receiving precursors with equal interference numbers promotes the growth of more complete forward flooding meshes at the network level. An expansive broadcast mesh improves a family's chances of either reaching its destination or finding a match. An example is shown in Figure 8.3 where different colors represent different statelet families, and the interference number of a statelet received by a node is written beside it using the color assigned to its family. A match is possible in Figure 8.3 between solid and dashed statelets of the same color. When the broadcast pattern of the blue precursor is satisfied before the red precursor at all nodes, a match results. When the broadcast patterns of precursors with equal interference numbers are arbitrarily satisfied, no match may be found, as shown in the illustrative example.


IntNo = 2
IntNo = 0
4
IntNo = 2
3
IntNo = 0
IntNo = 2
IntNo = 1
IntNo = 1
IntNo = 1
link
IntNo = 1
2
IntNo = 1
IntNo = 2
1
IntNo = 0
match
IntNo = 2
IntNo = 0
!
4
IntNo = 2
3
IntNo = 0
IntNo = 2
IntNo = 1
IntNo = 1
IntNo = 1
IntNo = 1
2
IntNo = 1
IntNo = 2
1
IntNo = 0

(a) Precursors with equal interference numbers are treated in an arbitrary order at a node

(b) The blue statelet is systematically treated before the red statelet at all nodes where they have equal interference numbers, i.e. node 1

Figure 8.3. Broadcasting precursors

Given a sorted precursor list in a node, the composite broadcast pattern over all families present at a node may be computed. When a SOURCE-FLOOD-INDICATOR is encountered, the node attempts to transmit from each relation it sources a number of statelets equal to the demand lost by that relation on the spares of each span. For example, if 5 and 2 units of demand are lost between nodes A - X and A - Y respectively, node A attempts to broadcast 5 statelets per span with source = A and destination = X, and 2 statelets per span with source = A and destination = Y. Similarly, because bi-directional flooding is being performed, node X attempts to broadcast 5 statelets per span with source = X and destination = A, and node Y attempts to broadcast 2 statelets per span with source = Y and destination = A.

The interference number of each statelet initiated at a source is determined by calculating the interference number of the span on which the statelet is transmitted and adding this value to the IIN. The interference number of a span is calculated by counting the number of precursors that need to transmit a statelet on that span in order to satisfy their target broadcast pattern, and subtracting from this sum the total number of spare links still available for use in restoration on that span. Ports locked by reverse linking are not available to carry forward flooding statelets, and are therefore not subtracted from the sum of statelets waiting to be rebroadcast. Since reverse-linking takes precedence over forward flooding, an incoming complement statelet at a tandem node is reverse linked, supplanting any outgoing forward propagating statelet occupying that port matched with the reverse linking statelet, before span interference numbers are calculated. Reverse linking is explained in greater detail later in this chapter.

The opportunity to establish restoration paths one hop long can arise when jointly optimizing the placement of working and spare capacity in a network. This would imply that a restoration path on the shortest route existed, but that the working demand was routed on a different one-hop route in parallel or via a longer route. Normal provisioning processes would put the working path on the shortest route. However in many of the combined capacity optimization designs presented in chapter 4 (i.e. design cases 5 and 6), the IP routed demand over a few working paths longer then the shortest route in order minimize the sum of working and spare capacity. When the working path routing is tailored to facilitate the target restoration mechanism, restoration paths one hop long are often possible, and OPRA is designed to exploit these opportunities when they present themselves.

The opportunity to restore a failed working path using a restoration path one hop long, as shown in Figure 8.4, is identified at a source node by comparing the destination of a forward flooding statelet this node proposes to transmit on a port to the NID present in the RxReg of that port. In the event the NID and destination fields are the same, a restoration path one hop long can be established, and only the source node with the larger ID, as determined by comparing this node's ID to that same NID field identified previously, initiates sending statelets on the connecting span. Establishing a restoration path one hop long represents the most efficient use of spare capacity in a network and is used to restore lost demand whenever possible.

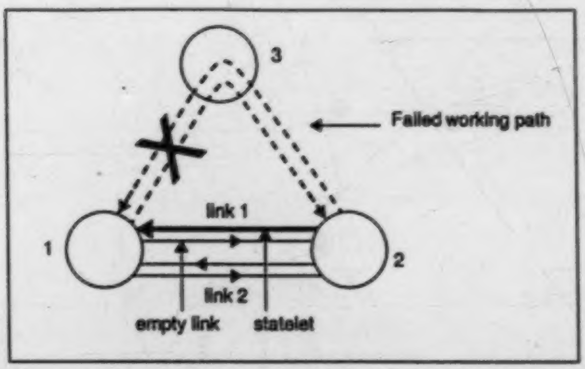


Figure 8.4. Establishing a restoration path one hop long

Only one of the nodes terminating a one-hop restoration path transmits statelets (node 2 in Figure 8.4) to avoid occupying more links than required, and to avoid entering an indeterminate state. For example, in Figure 8.4, if node 1 transmits a statelet on link 2, and node 2 transmits a statelet on link 1, two potential restoration paths may be identified simultaneously. The one that should be chosen to restore the single failed working path is indeterminate. Therefore only node 2 transmits a statelet on the span connecting nodes 1 and 2.

For a statelet entered into the list of precursors, OPRA attempts to transmit one statelet on one link in each outgoing span, provided that span doesn't contain an incoming forward flooding statelet from the same family as the precursor. OPRA doesn't attempt to transmit a statelet on a span which contains an incoming forward flooding statelet from the same family so that a precursor's broadcast mesh expands in a direction away from that precursor's source. As shown in Figure 8.5, whether the statelet on span 2 or 3 is chosen to be the precursor for the red statelet family at node 3, no branch should be established on spans 2 or 3, because it would grow the red broadcast mesh towards node 1. When this basic target broadcast pattern cannot be satisfied at a node for all families, the multi-family flooding pattern at a node is determined by competition among statelets based on their interference numbers.

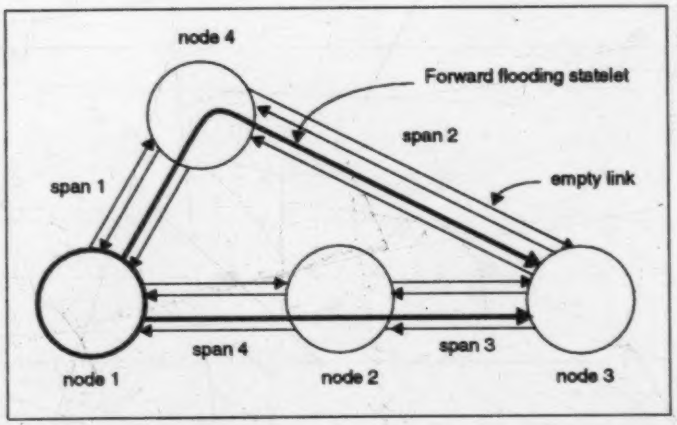
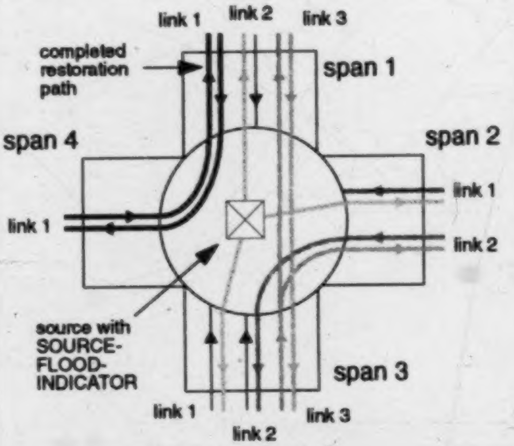


Figure 8.5. Growing a statelet's broadcast mesh away from the source.

In a span where all links already bear active outgoing statelets, competition among statelets may require that some families lose their outgoing statelet in that span. When a precursor with a lower interference number cannot access a span filled with statelets from other families, the precursor takes over the link occupied by the statelet whose interference number is the highest among those statelets present on the outgoing links of that span. The broadcast pattern for each precursor is adjusted until every precursor is either fully satisfied or is partially satisfied to the greatest extent possible, consistent with its overall rank in the list of precursors.

Every time the broadcast pattern is revised, the basic flooding pattern per statelet is applied, working up from those precursors with the lowest interference numbers to the highest, applying no statelet in a span that is already full. An example is shown in Figure 8.6. The patterns are graphically complex but the underlying principle is that maximal, mutually consistent flooding patterns are provided for multiple concurrent flooding processes, one per primary statelet emitted by a source. Their multilateral coordination at every node is on the basis of an incoming statelet's interference number. Statelet arrivals which are not of the lowest interference number for that family are ignored.


link 1
link 2
link 3
completed restoration path
span 1
span 4
span 2
link 1
link 2
link 1
source with SOURCE-FLOOD-INDICATOR
span 3
link 1
link 2
link 3

| statelet | | | Precursor Ranking | IntNo |
|---|---|---|---|---|
| Span | Link | In/Out | | |
| 1 | 1 | In | Completed Restoration Path | |
| | | Out | | |
| | 2 | In | empty | null |
| | | Out | 1 | 2 |
| | 3 | In | 4 | 4 |
| | | Out | 2 | 3 |
| 2 | 1 | In | 5 | 5 |
| | | Out | 1 | 1 |
| | 2 | In | 3 | 2 |
| | | Out | 2 | 2 |
| 3 | 1 | In | empty | null |
| | | Out | 1 | 1 |
| | 2 | In | empty | null |
| | | Out | 3 | 3 |
| | 3 | In | 2 | 1 |
| | | Out | 4 | 5 |
| 4 | 1 | In | Completed Restoration Path | |
| | | Out | | |

| Span/IIN | Interference Number |
|---|---|
| IIN | 0 |
| 1 | 4 precursors - 2 spares = 2 |
| 2 | 3 precursors - 2 spares = 1 |
| 3 | 4 precursors - 3 spares = 1 |
| 4 | Unavailable, already cross-connected |

Figure 8.6. Computing the composite broadcast pattern at a node

Finally, before the broadcast pattern at a node is computed, the *Broadcast* state checks whether a precursor is able to form a match. The details of recognizing a match are covered in detail later in this chapter.

The procedures executed upon transiting state 12 are summarized below:

1. Compile a list of all precursors present at this node. The precursor for a given statelet family has the lowest interference number of all forward flooding statelets belonging to the same statelet family. If two or more statelets

have the same minimum interference number, the one with the lower repeat count is the precursor.

2. Order the precursor list from lowest to highest interference number, using a precursor's source and destination ID to resolve ties.
3. Check for a match.
4. Compute the composite broadcast pattern at this node ensuring that:
    a. opportunities to establish a restoration path one hop long are exploited,
    b. no statelet is transmitted on a span containing a forward flooding statelet from that same family, and
    c. the broadcast pattern for each precursor is either fully satisfied, or partially satisfied, to the greatest extent possible, consistent with that precursor's overall rank in the list of precursors.

### 8.3.2 Event Parsing at a Tandem Node (State 1, *Tandem_Node*)

Whenever a statelet destined to cause a rebroadcast on the same family to another node is received, the rules and functions pertaining to updating the broadcast pattern of a statelet as explained in this section apply. Unlike the functions required to compute the composite broadcast pattern at a node, updating the broadcast pattern at a node by definition here does not involve transmitting new forward flooding statelets on links. Updating the broadcast pattern of a statelet is limited to pruning the branches of a precursor, moving the root of a precursor, reverse linking a complemented restoration statelet, and/or transmitting a loop-back test signal. Which function needs to be performed depends on the nature of the incoming statelet received. A tandem node may receive forward flooding, reverse linking, complemented, confirmation, and null statelets. Each of these statelets must be processed differently, and determining the nature of a statelet to facilitate its processing is the responsibility of the tandem node state.

Any time a statelet is received for which this node is not the destination, the transition from idle to tandem node occurs. Upon entering the tandem node state, OPRA determines whether a null statelet was received and whether a statelet existed previously on the port which issued this interrupt. If the port of the received statelet is non-null, it is put into the list of all non-null statelet's received by the node, *rec_set*.

Before exiting state 1, OPRA determines whether a precursor previously resided on the interrupting port by searching the *assoc_span* and *assoc_port* fields of each PSReg for a pointer to the interrupting port. To ensure OPRA functions correctly without acknowledging every interrupt received, rather than searching this node's precursor list, which may be outdated from the time it was created in the *Broadcast* state, all of the PSRegs at a node are searched for a branch rooted in the interrupting port. If any other port points to the interrupting port, there was a precursor on the port which issued the interrupt, and a branch from the precursor exists on the port whose *assoc_span* and *assoc_port* fields point to it.

The procedures executed upon transiting state 1 are summarized below:

1. Determine whether a statelet existed previously at the interrupting port, and if a null statelet now exists.
2. Enter any new incoming non-null statelet into *rec_set*.
3. Determine if the incoming statelet overwrote a precursor from another family. If so, record the source, destination, and index of that precursor.

Three of the five possible transition conditions leaving the tandem state belong conceptually to the stage described as bidirectional selective forward flooding, acting in a manner similar to that of a Sender node as defined in the SHN [25]. Processing these three events requires knowing whether a statelet existed previously at the interrupting port, and if a null statelet now exists. The first of these transitions occurs when a new non-null statelet arrives at a tandem node, and there was no prior statelet in that port.

The second transition condition occurs when a non-null statelet overwrites another non-null statelet at a tandem node. The third transition condition is satisfied when a null statelet overwrites a non-null statelet, i.e. a statelet vanishes. The destination states of these three transitions, states 2, 3, and 4 respectively, are discussed next.

### 8.3.3 Arrival of a New Restoration Statelet at a Tandem Node (State 2, *New_Statelet*)

When a non-null statelet is received at a tandem node on a previously empty port, state 2 named *New_Statelet* in Figure 8.1 is entered. Upon entering *New_Statelet*, OPRA determines whether the new statelet is better than any other statelet belonging to the same index-source-destination family. Of all the forward flooding statelets received at a node belonging to the same family, the statelet which:

1. doesn't trace a cyclical path starting and ending at its own source node, as detected by comparing the incoming statelet's source field to this node's ID,
2. hasn't exceeded the repeat limit,
3. doesn't occupy a port which this node is using to establish a restoration path one hop long,
4. doesn't belong to the family of a statelet already complemented, and
5. has the lowest interference number,

is considered best. If two or more statelets satisfy these conditions, the statelet with the lowest repeat count is chosen.

Conditions 1, 2, and 5 do not require further explanation; however, conditions 3 and 4 need some clarification. The third condition is required to ensure restoration paths one hop in length are used whenever possible. When establishing a restoration path one hop in length, as shown in Figure 8.4, only the node with the larger ID initiates sending statelets on the connecting span. However, the node with the smaller ID, node 1 in Figure 8.4, is allowed to relay a statelet from another relation over link 1 on the connecting span, until the statelet initiated by the adjacent node, node 2, is received. Therefore node 2 may momentarily receive a spurious statelet from another relation on a port which it is using to establish a restoration path one hop long.

The fourth condition is required to ignore a late-arriving forward flooding statelet from a family which either has already completed a restoration path, or is in the process of reverse linking or confirming a restoration path. Such families are said to be complemented locally at a node. While some late-arriving statelet may have a lower interference number than an existing complemented statelet from the same family, that late statelet is not allowed to interrupt its own families reverse linking or confirmation process. Considering that any statelet may establish a match and initiate reverse linking so that the restoration time as well as the amount of spare capacity used to restore a failure is minimized, OPRA prevents late-arriving forward flooding statelets from supplanting early arriving statelets which are already complemented locally at a node.

When a new statelet is the best precursor at a tandem node, its port becomes the precursor of that family. If a precursor existed prior to the arrival of a new best precursor, the old broadcast pattern is re-rooted onto the new precursor. Otherwise the new statelet waits until the broadcast state computes the composite broadcast pattern for this node.

The function of re-rooting a precursor is performed by updating the *assoc_span*, and *assoc_port* fields of all the branches of the old precursor to point to the new precursor. As shown in Figure 8.7, moving a precursor's root requires:

1. finding and cancelling any branches (statelets) from the old precursor lying in the same span as the new precursor leftover from the old broadcast pattern, and
2. finding all elements of the existing broadcast fan and swinging them to the new root.

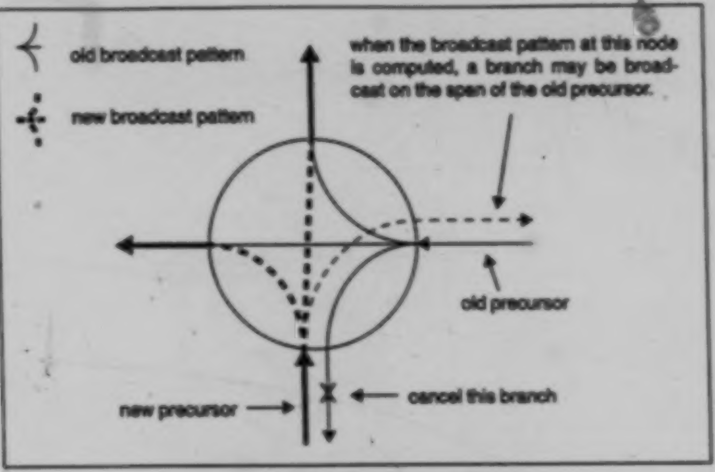

old broadcast pattern
new broadcast pattern
when the broadcast pattern at this node is computed, a branch may be broadcast on the span of the old precursor.
old precursor
new precursor
cancel this branch

Figure 8.7. Re-rooting a precursor

In general, whenever a precursor is overwritten, before its branches are cancelled, OPRA checks whether those branches may be rooted onto a new precursor from the same family. This is done because a precursor may disappear at a node, due to the collapse of its broadcast mesh elsewhere in the network, or shift from one port to another at the same node during the normal course of forward flooding, in which case the broadcast mesh of that statelet family should be rerooted rather than cancelled. Consider Figure 8.8, given that the red statelet overwrote the blue precursor on link 4 at node A, and the blue statelet overwrote the red precursor on link 1 at node A, instead of cancelling the forward flooding statelets transmitted on links 3 and 5, it is advantageous to reroot the blue broadcast pattern onto link 1, and the red broadcast pattern onto link 4. When the broadcast pattern is rerooted instead of cancelled, reverse linking is continued at node A rather than stopped, which is desirable because it may lead to the completion of a restoration path.

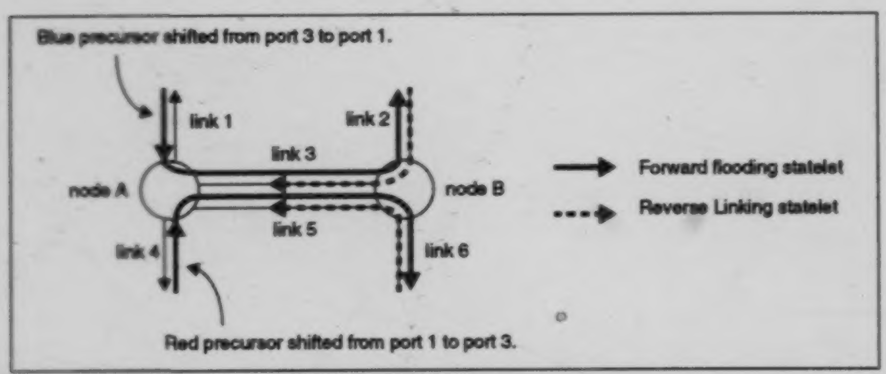

Blue precursor shifted from port 3 to port 1.
link 1
link 2
link 3
node A
node B
link 5
link 4
link 6
Forward flooding statelet
Reverse Linking statelet
Red precursor shifted from port 1 to port 3.

Figure 8.8. Rerooting a broadcast mesh after a precursor has changed positions

The procedures executed upon transiting state 2 are summarized below:

1. Determine if the statelet received should become the precursor of that family. If so, reroot the branches of that family onto the interrupting port.

### 8.3.4 Overwriting a Restoration Statelet at a Tandem Node (State 3, *Statelet_Ovwrt*)

When a non-null statelet overwrites another non-null statelet at a tandem node, state 3, *Statelet_Ovwrt* in Figure 8.1, is entered. To facilitate processing, the overwrite event is put into one of four classes:

1. a statelet overwrites an existing precursor from its own family, or
2. a statelet overwrites a non-precursor statelet on its own family, or
3. a statelet overwrites an existing precursor from another family, or
4. a statelet overwrites a non-precursor statelet from another family.

The first action associated with case 1 determines if the statelet received is better qualified to be that family's precursor than any other statelet present belonging to the same family. If the statelet received is not the best, a new precursor is sought amongst the other incoming statelets from the same family to replace the one overwritten. The old broadcast pattern is then re-rooted onto the new precursor. If the repeat count of the new incoming statelet exceeds the repeat limit, and no new precursor is found to replace the one overwritten, any branches of the old precursor are cancelled. In the event a statelet overwrites an existing precursor from its own family on a connected port part of a reverse linking process, and the newly received statelet is qualified as best, the entry in that port's TxReg register is sustained with updated fields only.

If a statelet overwrites a statelet from its own family which was not the precursor, the actions required to process the second case of statelet overwrite events are followed. If the newly received statelet is found to be the best of all statelets received for that family, the current precursor of that family is found. Because the new statelet overwrote a statelet which was not the precursor of that family, the precursor for that family must reside on another port. After finding that port, the family is re-rooted onto the new statelet as precursor.

In the third case, if a statelet overwrites an existing precursor from another family OPRA attempts to find a new precursor to replace the one overwritten. If a new precursor is found, the statelet family for that index-source-destination label is rerooted, otherwise the branches of the prior broadcast pattern from that family are cancelled. Then, if the newly received statelet is best for *its* family, that family is rerooted onto the interrupting port.

In case 4, the actions required to process a statelet which overwrites a statelet from another family, which was not a precursor, are followed. If the statelet received is best, the statelet family of the received statelet is rerooted onto the interrupting port, otherwise nothing is done.

### 8.3.5 Disappearance of a Restoration Statelet at a Tandem Node (State 4, *Statelet_Vanish*)

When a non-null statelet vanishes at a tandem node, state 4 named *Statelet_Vanish* in Figure 8.1 is entered. Upon entering *Statelet_Vanish*, the port issuing the interrupt is removed from the list of all non-null statelets received by a node, *rec_set*. If the statelet which vanished was a precursor, OPRA looks within the other statelets on that family to see if another has become the valid precursor. If a new precursor is found, the old broadcast pattern is re-rooted onto the new precursor, otherwise the branches of the old broadcast pattern are cancelled.

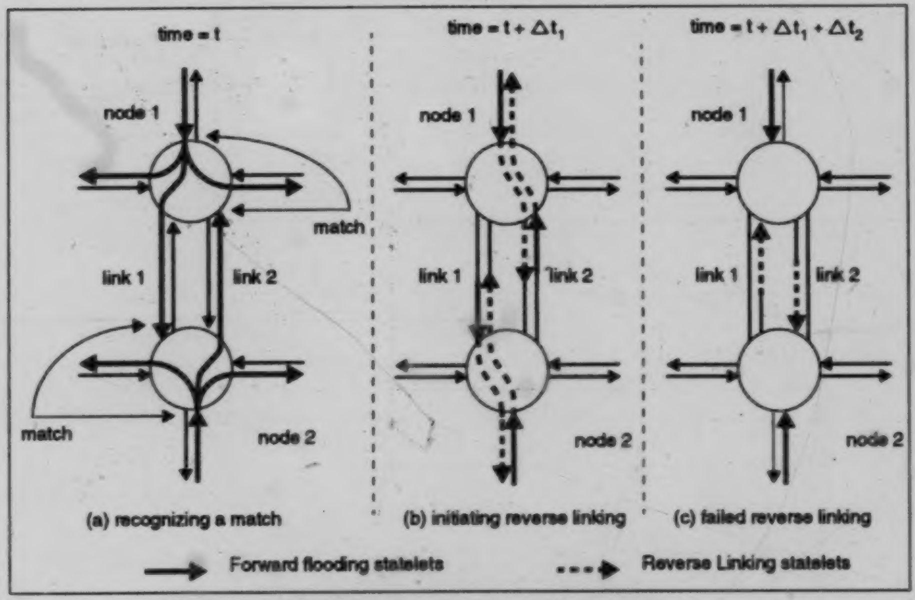
## 8.4 Recognizing a Match

When two forward flooding statelets meet at a tandem node, and the source of one statelet corresponds with the destination of the other, and vice-versa, an event known as a match occurs. While matches occur relatively frequently during the dynamic phase of restoration, initiating reverse linking as a result is controlled to minimize starting those reverse linking processes which are destined to fail, and maximizing those which lead to the creation of a near optimal restoration pathset.

Before the composite broadcast pattern for a node is computed in the *Broadcast* state, the possibility of forming a match between two precursors is evaluated. The procedure responsible for identifying a match is placed in the *Broadcast* state so it can access the same list used to compute the composite broadcast pattern for the node at the time it is compiled and therefore up-to-date. Every statelet in the precursor list is checked to see whether it is a match to the new statelet. When two forward propagating statelets in the precursor list establish a match locally at a node the following conditions must be satisfied before initiating reverse linking:

1. The NID number of the node where the match occurred must be less than the NID number of an adjacent node which is transmitting one of the matched precursors and receiving a forward flooding statelet from the same relation. (The NID is borne in the statelet so this arbitration is simple to effect.)
2. The node where the match occurred must not be sourcing a statelet whose destination is the adjacent node on either of the matched ports.
3. The matched statelets must not reside on the same span.

The first condition is required when a match occurs between a forward flooding statelet just received and a precursor that was previously broadcast. Completing a match involves cancelling the branches of a precursor and transmitting a reverse linking statelet on the port of the forward flooding statelet just received. As shown in Figure 8.9, if both nodes 1 and 2 simultaneously complete a match, node 1 transmits a reverse linking statelet on link 2 and cancels the forward flooding statelet on link 1, while node 2 transmits a reverse linking statelet on link 1 and cancels the branch on link 2. Subsequently the forward flooding statelet on link 2 at node 1 disappears and the forward flooding statelet at node 2 on link 1 disappears. When the forward flooding statelet on a matched port disappears, the complemented as well as the forward flooding statelets on the matched ports are cancelled as shown in Figure 8.9 (c), terminating the construction of a possible restoration path. To avoid this, only the node with the smaller ID is allowed to acknowledge a match, while the node with the larger ID simply waits for the arrival of the reverse linking statelet on the matched port.


time = t
node 1
match
link 1
link 2
match
node 2
time = t + Δt₁
node 1
link 1
link 2
node 2
time = t + Δt₁ + Δt₂
node 1
link 1
link 2
node 2
Forward flooding statelets
Reverse Linking statelets

(a) recognizing a match (b) initiating reverse linking (c) failed reverse linking

Figure 8.9. Problems when two tandem nodes complete a match simultaneously

If node 1 were a source rather than a tandem node, node 2 could safely complete a match because node 1 would not cancel any forward flooding statelet that it sources, even if it received a forward flooding statelet on link 2 from node 2.

The second condition is required in the event a restoration path one hop long can be established. As shown in Figure 8.10, the match at node 2 is not recognized because it would require transmitting complemented red statelets on spans 1-2 and 2-3, thereby overwriting the blue statelet and eliminating the opportunity to create a restoration path one hop long. Restoration paths one hop long represent the most efficient use of spare capacity and are therefore given priority over longer restoration paths by this mechanism. This also contributes to the principle of protecting each nodes adjacent spans for its own demands to "escape" through the wider network, i.e. avoiding end-node region traversal by other families.

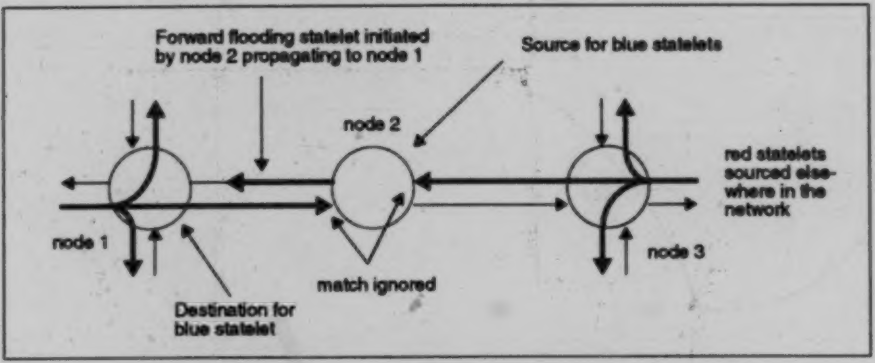
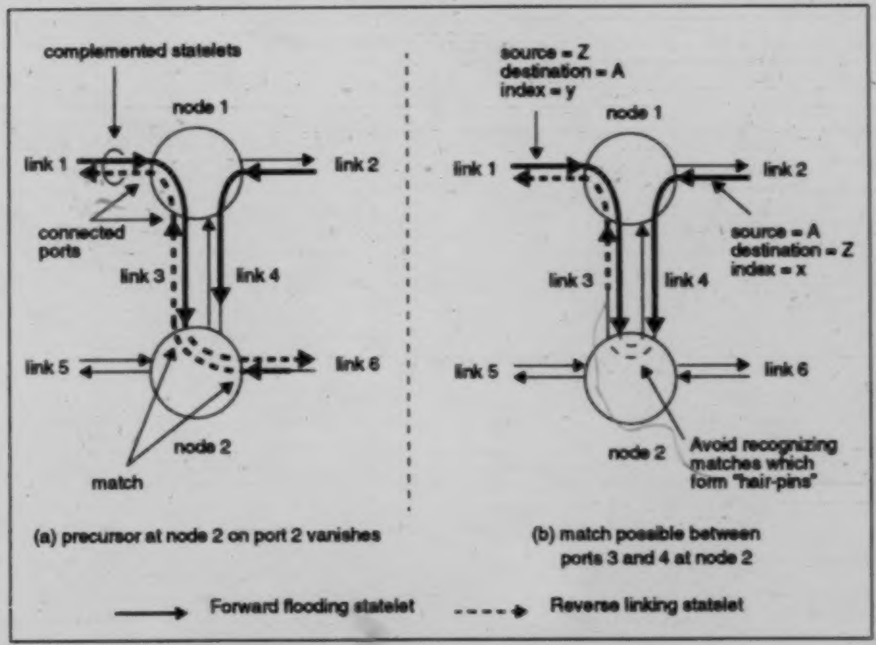

Forward flooding statelet initiated by node 2 propagating to node 1
Source for blue statelets
node 2
red statelets sourced elsewhere in the network
node 1
node 3
match ignored
Destination for blue statelet

Figure 8.10. Ignoring a match when a node is attempting to establish a restoration path one hop long

The third and final condition avoids completing a match between two statelets received on the same span. Such a match could create an inefficient restoration path which forms a "hair-pin" turn as shown in Figure 8.11. In the event a forward propagating statelet arrives at node 2 on link 4 shortly after the precursor on link 6 vanishes, a match between links 3 and 4 is possible as shown in Figure 8.11 (b). However, recognizing the match at node 2 would form a hair-pin turn which wastes spare capacity. Instead, OPRA ignores the match at node 2 and waits for the complemented statelet on link 1 and the forward flooding statelet on link 3 to be cancelled when the complemented statelet on link 3 disappears, before re-attempting a match between the forward flooding statelets received on links 1 and 2.


complemented statelets
node 1
link 1
link 2
connected ports
link 3
link 4
link 5
link 6
node 2
match
source = Z
destination = A
index = y
node 1
link 1
link 2
source = A
destination = Z
index = x
link 3
link 4
link 5
link 6
node 2
Avoid recognizing matches which form "hair-pins"
Forward flooding statelet
Reverse linking statelet

(a) precursor at node 2 on port 2 vanishes

(b) match possible between ports 3 and 4 at node 2

Figure 8.11. Avoiding "hair-pins" when forming restoration paths

When two matched precursors satisfy the four conditions explained above, they are removed from the precursor list. All of their branches, except those on the matched ports, are cancelled, and a reverse linking statelet is transmitted at each matched port, i.e. their broadcast mesh collapses onto the matched ports.

Transmitting reverse linking statelets requires that the complemented field of the reverse linking statelet be set to the index of the forward flooding statelet received on the same port. Because the direction of a reverse linking statelet and that of its precursor are the same, the source, destination, and index of a reverse linking statelet and its precursor are the same.

The repeat count and interference number of a reverse linking statelet are needed to record the length and interference number of an entire restoration path from source to destination. Therefore, the repeat field of a reverse linking statelet continues to accumulate as it is repeated by a tandem node, and the IntNo field is set to the sum of the interference numbers of the matched precursors. For example, if two matched precursors, x and y, have interference numbers of 3 and 7 respectively, the IntNo field of both reverse linking statelet's transmitted is set to 10. Unlike the repeat counter, the IntNo of a reverse linking statelet is never altered once set during a match by subsequent tandem nodes.

After a match is completed, no other statelet may be transmitted on either of those ports. The matched ports are locked by setting the *activated* field of the PSReg to true. While statelets may not be transmitted on connected, or *activated* ports, new statelets may be received, including null statelets. Often a forward flooding statelet will disappear or be overwritten on a connected port because it was supplanted or eliminated elsewhere in the network. When a new statelet is received, the *activated* field is usually set to false. The resetting of a port's *activated* field to false is explained in more detail in the following section.

Though resetting a port's *activated* field to false terminates reverse linking and stops the construction of a restoration path, generally only those forward flooding statelets associated with suboptimal paths are overwritten. OPRA will abandon suboptimal paths in favour of new and better restoration paths until those forward flooding statelets with the lowest interference number, which represent the best paths are sustained, allowing reverse linking to complete.

## 8.5 Reverse Linking

Reverse linking begins after a match is recognized and collapses the mesh of a matched forward flooding statelet family in the network onto a subset of links that trace a path between the matched statelet's source and destination. The completion of reverse linking identifies a path through the cooperating tandem nodes. The successful completion of the confirmation process for this path establishes one restoration path available to restore one unit of lost demand.

While reverse linking is usually initiated by a match at a tandem node, the destination of a forward flooding statelet may also initiate reverse linking. It is possible that a forward flooding statelet will not collide with another statelet from the same relation travelling in the opposite direction before that statelet reaches its destination. Whether a forward flooding statelet finds its destination or a match, a path is identified, and construction of a possible restoration path is continued by initiating reverse linking.

When a reverse linking statelet arrives at a tandem node, OPRA takes the following steps:

1) all rebroadcast statelets emitted by the tandem node having the same source-destination-index family as the reverse linking statelet are deleted,
2) a logical connection between the port where the reverse-linking statelet and the port of the precursor for that family is presently found is formed by setting the *activated* field of the PSReg to true,
3) a new complemented statelet is originated in the transmit direction at the port containing the precursor, and
4) general reorganization of the forward flooding statelets at a node occurs to effect an optimized reallocation of new ports available for restoration amongst the set of statelet families still outstanding at this node. The composite broadcast pattern is revised to take all effects of statelet removals into account in the revised pattern.

After transiting one or more tandem nodes, a reverse linking statelet arrives back at the source of the forward flooding statelet with which it is paired. A complete bidirectional path is then known to have been traced between the source and destination, although no one node knows the overall routing. However, whether the path will be sustained and utilized to restore lost demand isn't known until the continuity test associated with the confirmation phase of the restoration process is performed. The following sections detail the functions associated with reverse linking. These functions are implemented using state 5, *Complement_Statelet*, shown in Figure 8.1.

### 8.5.1 Receiving a Reverse Linking statelet (State 5, *Complement_Statelet*)

When a complemented statelet is received at a tandem node the transition condition leading from state 1 to state 5 in Figure 8.1 is satisfied. The first action performed upon entering the *Complement_Statelet* state determines if the complemented statelet received over-wrote a precursor from another family. If so, a new precursor is sought to replace it.

Then OPRA determines whether reverse linking should be stopped or allowed to continue. Reverse linking is only allowed to continue if the forward propagating statelet with which this statelet is matched has not disappeared or been supplanted locally. Referencing Figure 8.12, to continue reverse linking at a tandem node the following conditions must be satisfied:

1) the source field of the forward flooding statelet transmitted on port 1 and received on port 3 must be equal to the destination field of reverse linking statelet on port 1,
2) the destination field of the forward flooding statelet transmitted on port 1 and received on port 3 must be equal to the source field of reverse linking statelet on port 1, and
3) the index of forward flooding statelet transmitted on port 1 and received on port 3 must equal the reverse linking indicator of the complemented statelet on port 1.

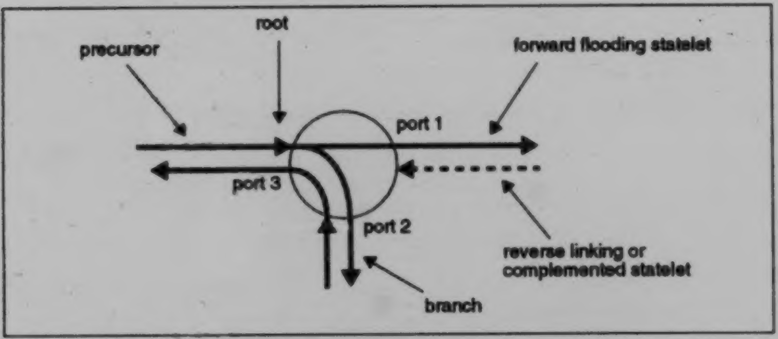

root
precursor
forward flooding statelet
port 1
port 3
port 2
reverse linking or complemented statelet
branch

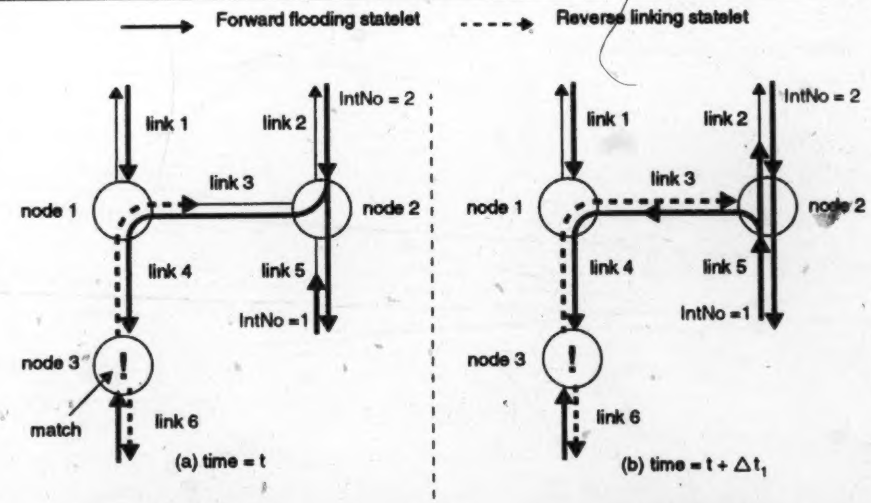
Figure 8.12. Receiving a reverse linking statelet

A by-product of these conditions is that the statelets on the receive and transmit links of a given port are never both reverse linking or forward flooding. If these conditions are satisfied, reverse linking continues, and the complemented statelet is further relayed back to the source of the forward flooding statelet, supplanting the blue statelet on port 3 in Figure 8.12, otherwise reverse linking is stopped.

To explain why reverse linking may stop, consider Figure 8.13, in which the blue forward flooding statelet is received at node 2 on link 5 before the reverse linking statelet is received on link 3. Since the interference number of the blue statelet received on link 5 is less than the interference number of the red precursor on link 2, and the *activated* field of link 3 is not set to true as shown in the table, the blue statelet on link 5 is allowed to supplant the red statelet on link 3. Consequently, when the complemented statelet is received on link 3 at node 2, reverse linking is stopped. Had the reverse linking statelet arrived before the blue statelet, node 2 would have relayed the reverse linking statelet along the path traced by the precursor on link 2 and logically connected links 2 and 3, setting the *activated* field of both ports to true, thereby preventing the blue statelet from transmitting any forward flooding statelets at node 2. When the *activated* field of a port is set to true, no statelet may supplant the statelet being transmitted on that port.

When the blue forward flooding statelet is received at node 1 on link 3, the red forward flooding statelet on link 4 is cancelled along with the red reverse linking statelet on link 3. Subsequently the broadcast patterns of the blue precursor on link 3 at node 2 and the green precursor on link 1 are satisfied to the greatest extent possible, consistent with the overall rank of the statelets.

By suspending the reverse linking process of those statelets with higher interference numbers, OPRA resolves itself into those restoration paths with the lowest interference number. Unless the reverse linking process of a forward flooding statelet with a large interference number is very quick, it may not succeed, as depicted in Figure 8.13. Given that all the branches of a broadcast mesh seek a match simultaneously, many reverse linking processes may be initiated at the same time for a given statelet family, each racing to collapse the broadcast mesh on itself. Only the reverse linking process which succeeds in cutting off all other branches in the broadcast mesh survives this race, and all other reverse linking processes from that family die. Note that this is a mechanism which helps self-allocate the available spare links amongst the relations to be restored.


Forward flooding statelet
Reverse linking statelet
link 1
link 2
IntNo = 2
link 3
node 1
node 2
link 4
link 5
IntNo =1
node 3
!
match
link 6
link 1
link 2
IntNo = 2
link 3
node 1
node 2
link 4
link 5
IntNo =1
node 3
!
link 6

(a) time = t

(b) time = t + $\Delta t_1$

Port Register Fields

| node | link | Tx/Rx | *source* | *dest.* | *index* | *rli** | *activated* |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Tx | null | null | null | null | false |
| | | Rx | E | F | i | null | |
| | 4 | Tx | A | B | x | null | true |
| | | Rx | *B* | *A* | *y* | *x* | |
| | 3 | Tx | *B* | *A* | *y* | *x* | true |
| | | Rx | A | B | x | null | |
| 2 | 2 | Tx | null | null | null | null | false |
| | | Rx | A | B | x | null | |
| | 5 | Tx | A | B | x | null | false |
| | | Rx | null | null | null | null | |
| | 3 | Tx | A | B | x | null | false |
| | | Rx | null | null | null | null | |
| 3 | 4 | Tx | *B* | *A* | *y* | *x* | true |
| | | Rx | A | B | x | null | |
| | 6 | Tx | *B* | *A* | *y* | *x* | true |
| | | Rx | A | B | x | null | |

rli: reverse linking indicator
bold: forward flooding statelet
italic plain: reverse linking statelet

Port Register Fields

| node | link | Tx/Rx | *source* | *dest.* | *index* | *rli** | *activated* |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Tx | null | null | null | null | false |
| | | Rx | E | F | i | null | |
| | 4 | Tx | A | B | x | null | true |
| | | Rx | *B* | *A* | *y* | *x* | |
| | 3 | Tx | *B* | *A* | *y* | *x* | true |
| | | Rx | A | B | x | null | |
| 2 | 2 | Tx | C | D | u | null | false |
| | | Rx | A | B | x | null | |
| | 5 | Tx | A | B | x | null | false |
| | | Rx | C | D | u | null | |
| | 3 | Tx | C | D | u | null | false |
| | | Rx | *B* | *A* | *y* | *x* | |
| 3 | 4 | Tx | *B* | *A* | *y* | *x* | true |
| | | Rx | A | B | x | null | |
| | 6 | Tx | *B* | *A* | *y* | *x* | true |
| | | Rx | A | B | x | null | |

rli: reverse linking indicator
bold: forward flooding statelet
italic plain: reverse linking statelet

Figure 8.13. Reverse linking

If the forward flooding statelet paired with this complemented statelet has disappeared or been supplanted, reverse linking is stopped, and nothing else is done. Otherwise the broadcast pattern of the precursor for the forward flooding statelet paired with the complemented statelet (e.g. the statelet received on port 3 in Figure 8.12) is pruned leaving only the branch on the port receiving the reverse linking statelet (e.g. the statelet transmitted on port 1 in Figure 8.12). Pruning the branches of the precursor collapses the broadcast pattern of the forward flooding statelet onto a single path and allows those statelet families still vying for complete rebroadcast at this node to extend their broadcast pattern. Reverse linking then transmits the complemented statelet on the port receiving the precursor (e.g. port 3 in Figure 8.12), supplanting a statelet from another family if necessary. Relaying a complemented statelet locks those ports involved in reverse linking (ports 1 and 3 in Figure 8.12) by setting the *activated* field of the PSReg to true. In the event a complemented statelet from another family is received on a connected port, the connection is broken by nulling the outgoing links of the connected ports and setting their *activated* field to false.

The procedures executed upon transiting state 5 are summarized below:

1. Determine if the incoming statelet overwrote a precursor from another family. If so, reroot the branches of that old precursor onto a new precursor; otherwise cancel its branches.
2. Determine whether reverse linking should continue. If so, collapse the broadcast mesh of the received complemented statelet onto the matched ports and continue reverse linking; otherwise, do nothing.

## 8.6 Restoration Path Confirmation

After successfully transiting one or more tandem nodes, a reverse linking statelet arrives at its destination, which is the source of the forward flooding statelet with which it is matched. In addition to complemented statelets, a destination node may also receive forward flooding statelets or confirmation statelets. A destination must therefore be able to process three different types of statelets: forward flooding, reverse linking, and confirmation statelets. State 9, *Destination_Node* in Figure 8.1 identifies the type of statelet received and performs functions common to the processing of all three. State 8, *Receive_Statelet*, performs functions specific to the processing of forward flooding and complemented statelets at a destination node. State 6, *Confirmation*, performs functions specific to the processing of confirmation statelets at a tandem node. State 10, *Restore_Signal*, performs functions specific to the processing of confirmation statelets at a destination node. The details of states 6, 8, 9, and 10 relating to the confirmation of a restoration path are discussed next.

### 8.6.1. The Arrival of a Statelet at a Destination Node (State 9, *Destination_Node*)

Whenever a statelet is received on an active port at a destination node, the transition condition leading from state 14 to state 9 in Figure 8.1 is satisfied. Upon entering this *Destination_Node* state the statelet is inserted into the list used to track all non-null statelets received at this node, *rec_set*. Next, OPRA determines whether the statelet received over-wrote a precursor from another family by searching the *assoc_span* and *assoc_port* fields of each PSReg for a pointer to the interrupting port. If a branch from an old precursor is found, a replacement precursor for it is sought. Provided a new precursor is found, the old broadcast pattern is re-rooted onto the new precursor, otherwise the branches of the old precursor are cancelled.

### 8.6.2 Processing a Forward Flooding or a Complemented Statelet at a Destination Node (State 8, *Receive_Statelet*)

Whenever a forward flooding or a complemented statelet is received at a destination node, state 8, *Receive_Statelet*, is entered. If the interrupting port contains a forward flooding statelet from an adjacent node the opportunity to establish a restoration path one hop long exists. In this event, only the source with the larger ID initiates sending statelets. The node with the smaller ID, i.e. this node, transmits a confirmation statelet to the adjacent node on the interrupting port and locks the port by setting the *activated* field of the PSReg to true. Establishing a restoration path one hop long represents the most efficient use of spare capacity, and the *Receive_Statelet* state checks if the interrupt was caused by a forward flooding statelet from an adjacent node first.

Whenever a destination receives a forward flooding or a complemented statelet from some node other than one adjacent to it, OPRA knows a path longer than one hop was traced. However, whether this path will be sustained is not known because only when all the ports along the path are locked (i.e. *activated* = true for each port) is it impossible to supplant those forward and backward propagating statelets that act as a bi-directional holding thread for that path. Because none of the ports along the path traced by a forward flooding statelet are locked until reverse linking is performed, the forward flooding statelet received by a source or destination may disappear. For example, when a forward flooding statelet arrives at a destination node and that statelet is supplanted at a tandem node elsewhere in the network, the forward flooding statelet will vanish shortly after being received. Furthermore, a complemented statelet received at an end-node will vanish or be overwritten when the complemented statelet propagating in the opposite direction fails to reach its destination. To sustain a path, both reverse linking statelets resulting from a match at a tandem node must reach their destinations.

To ensure that all the ports along the path traced by a reverse linking statelet are locked, the node with the smaller ID initiates a loop-back test by transmitting a statelet to the far-end node with the confirmation field set to a unique value. If all the tandem nodes along a path are connected, the far-end node receives the confirmation statelet and sends it back to the originating node. When the source and destination of a restoration path receive a confirmation statelet, a complete restoration path which will not disintegrate is identified.

The *Receive_Statelet* state is responsible for initiating the loop-back test to confirm the lock-in of a restoration path. When a complemented or forward flooding statelet is received at a destination node on a port which is transmitting a statelet from the same relation, and the receiving node's ID is less than the ID of the node terminating the far-end of the path, a confirmation statelet is transmitted on the interrupting port. Otherwise, the *activated* field of the interrupting port is set to true and the node waits for a confirmation statelet from the node with the smaller ID. If all the ports along the statelet's path are connected, the confirmation statelet will appear shortly. Otherwise the forward flooding or complemented statelet in that port will disappear.

Both complemented and forward flooding statelets received at a destination node on a port that is transmitting a statelet from the same relation are treated identically because the port which initiated the statelet will be locked. As shown in Figure 8.14, even though the statelet received at node 2 is not complemented, ports 1 and 2 at node 1 are connected because a match condition exists. The statelet received at node 2 was not complemented because the match at node 1 was recognized after transmitting the forward flooding statelet on port 2.

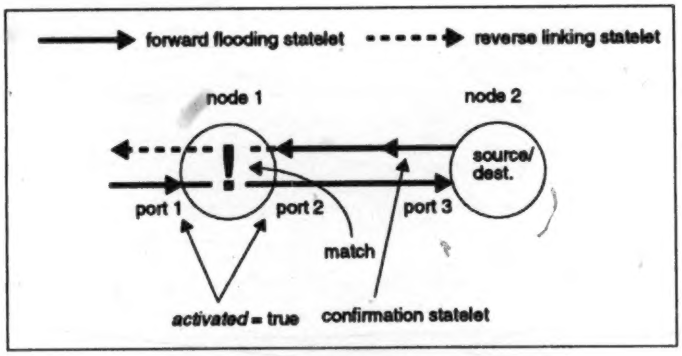
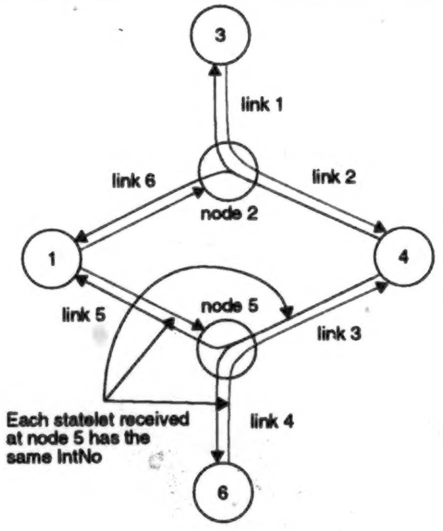


Figure 8.14. Initiating a loop-back test

However, when a forward flooding statelet is received at a destination node on a port that is not transmitting a statelet from the same relation, the port which initiated the statelet will not be locked and it is not known whether there is still any demand to be restored. If any lost demand remains to be restored from the relation identified by the forward flooding statelet just received, this node sources a statelet from that relation on the interrupting port.

Originally OPRA was designed such that forward flooding statelets received at a destination node on a port that was not transmitting a statelet from the same relation were ignored. However, it was found that anchoring a statelets broadcast pattern at the source and destination did not yield the best pathsets. As shown in Figure 8.15, unless a destination node is allowed to transmit statelets in response to the receipt of a forward flooding statelet, the network may reach a state in which no matches are found and no restoration paths are formed. Therefore, destinations were designed to transmit a complemented statelet from the same relation as the received statelet on an empty port, or a port occupied by a statelet from another relation, even if the node was already transmitting a number of statelets equal to the demand lost by that relation on that span.



| relation | lost demand | colour |
|---|---|---|
| 3 - 6 | 1 | green |
| 1 - 4 | 1 | blue |
| 1 - 3 | 1 | red |

Note: The interference number of a statelet is set to zero at the source, and when two or more statelets arriving at a node have the same interference number, the blue statelet's broadcast pattern is satisfied first, the green statelet's broadcast pattern is satisfied second, and the red statelet's broadcast pattern is satisfied last.

Figure 8.15. Problems associated with anchoring a statelet's broadcast mesh at the source

When the ID of the destination node receiving the forward flooding statelet is greater than the ID of the statelet's source, a complemented statelet is transmitted on the interrupting port, otherwise a confirmation statelet is transmitted. A confirmation statelet may be transmitted immediately in some cases because not only will a confirmation statelet ensure that all the ports in a path are connected, it will also collapse a forward flooding statelet's broadcast pattern and set a port's *activated* field to true whenever possible.

The procedures executed upon transiting state 8 are summarized below:

1. Establish a restoration path one hop long whenever possible.
2. If this node is transmitting a forward flooding statelet on the interrupting port from the same relation as the statelet received, and this node's ID is smaller than that of the source of the statelet received, initiate a loop-back test. Otherwise only lock the interrupting port by setting its *activated* field to true.
3. If this node is not transmitting a forward flooding statelet on the interrupting port from the same relation as the statelet received, and unrestored demand from the relation identified by the received statelet exists, initiate a loop-back test (transmit a complemented statelet) on the interrupting port if this node's ID is smaller (larger) than that of the source of the statelet received.

### 8.6.3 Processing a Confirmation Statelet at a Tandem Node (State 6, *Confirmation*)

When a confirmation statelet is received at a tandem node the transition condition leading from the *Tandem* state to the *Confirmation* state in Figure 8.1 is satisfied. If the interrupting port is not connected (i.e. *activated* = false), OPRA determines whether or not the statelet transmitted on that port, and the precursor to the statelet transmitted on that port, are in a complement relationship to the confirmation statelet just received. Similar to the processing of a complemented statelet at a tandem node, the confirmation statelet received is relayed on the port containing the precursor to the statelet transmitted on the interrupting port, thereby continuing the loop-back test if

1) the source of the forward flooding, or complemented statelet, transmitted on the interrupting port and that of its precursor are the destination of the confirmation statelet,
2) the destination of the forward flooding, or complemented statelet, on the interrupting port and that of its precursor are the source of the confirmation statelet, and
3) the index of forward flooding, or complemented statelet, transmitted on the interrupting port and that of its precursor are equal to the reverse linking indicator of the confirmation statelet,

Relaying a confirmation statelet at a tandem node on a port which wasn't previously locked by reverse linking requires:

1) collapsing the broadcast pattern of the precursor for the forward flooding statelet residing on the port receiving the confirmation statelet, and
2) locking those ports terminating the confirmation statelets by setting their *activated* fields to true.

Instead of simply checking that the port receiving a confirmation statelet is connected, i.e. *activated* = true, before continuing a loop-back test, attempting to collapse the broadcast mesh of a confirmation statelet as described above allows a destination node to skip the last logical step of local reverse linking and transmit a confirmation statelet immediately upon receiving a forward flooding statelet.

Whether a loop-back test is continued or not, the *Confirmation* state always determines if the confirmation statelet received over-wrote a precursor from another family. If so, a new precursor is sought to replace it. If a new precursor is found, the old broadcast pattern is re-rooted onto the new precursor, otherwise the branches of the old precursor are cancelled.

### 8.6.4. Traffic Substitution (State 10, *Restore_Signal*)

When a node to be restored, i.e. a source/destination node, receives a confirmation statelet, the transition condition leading from state 9 to state 10 in Figure 8.1 is satisfied. If this node is the destination of the confirmation statelet, a loop-back is performed and the confirmation statelet returned to its source. The return path of the confirmation statelet traverses the same set of ports traversed while propagating from source to destination.

If this node is the source of the confirmation statelet, and the *confirmation* field of both the transmitted and received confirmation statelets on the interrupting port are equal, a successful loop-back test is complete and a single unit of demand may be restored over the path traced by the confirmation statelet. The forward and backward propagating statelets that remain for a given family after successfully completing a loop-back test act as a bidirectional holding thread for that restoration path, sustaining it as long as the source and destination do not cancel them.

After successfully completing a restoration path, the source of the confirmation statelet bridges one of the failed transport signals onto the restoration path. When a unit of lost capacity is restored, a source/destination node increases the IIN of all statelets it sources in ongoing broadcasts by IIN_STEP. Increasing the IIN of those statelets sourced by this node increases the likelihood that other demand pairs which have not restored as much lost capacity as this node will complete one of their restoration paths. In a network which is not 100% restorable, or in the case of node recovery, this helps to achieve an overall pattern of individual restoration levels on each relation that is more proportional to the network's pre-failure connectivity, as shown in the results presented at the end of this thesis. Given that the networks used to test OPRA were 100% restorable against all single span cuts, IIN_STEP was set to zero when testing OPRA's ability to restore all single span cuts.

After restoring a unit of lost capacity, the count of working demands to be restored by this relation is decreased. When all of the demand between this node and a far-end node is restored, all of the outstanding initial broadcast statelets for that relation are cancelled.

## 8.7 Terminating a Restoration Event

OPRA synthesizes restoration paths until 100% restoration is achieved, at which time it inherently stops itself, or until the time-out interrupt is issued, signalling the termination of a restoration event. In the event a network spared for less than 100% restorability, OPRA will reach a static best effort restoration equilibrium, and the time-out interrupt is needed to "tidy-up" a network by cancelling any forward flooding statelets which have not been confirmed. Because OPRA is designed to restore a failure within two seconds, a conservative time-out limit of 3 seconds was used in all tests.

### 8.7.1 Cancelling Outstanding Statelets (State 13, *TimeOut*)

Each node starts its own timer upon entering a restoration event. Given that not all nodes are activated simultaneously, nodes are also not deactivated simultaneously. When a node times-out, the transition condition leading from the idle state to the *TimeOut* state is satisfied. Upon entering *TimeOut* state all of the statelets this node is sourcing which have not successfully completed a loop-back test are cancelled.

## 8.8 Network Level View

The seven conceptual processes just described actually occur concurrently at different stages and times at all nodes involved in a restoration event. Forward flooding proceeds simultaneously for several statelet families while other statelet families may be reverse linking. Each family of statelets, comprising the network-wide set of selective rebroadcasts resulting from each primary statelet issued by a source, competes against others to expand in the forward flooding process. Successful statelet families collapse in the process of reverse linking.

During forward flooding the precursor of each statelet family that has the lowest interference number over all statelets present at a node asserts its target broadcast pattern fully. In a span where all links already bear active outgoing statelets, the link occupied by the statelet whose interference number is the highest is supplanted. Each node acting as a tandem continues to adjust the broadcast pattern for each precursor in response to incoming statelet changes occurring at its ports so that every precursor at the node is either fully satisfied, or is partially satisfied to the greatest extent possible, consistent with the precursor's overall rank in terms of the interference numbers of incoming statelets and the spans in which it and other precursors lie. When the precursor location for a statelet shifts, the broadcast pattern for that statelet family is rerooted onto the new precursor and the composite broadcast pattern adjusted. When a new statelet appears at a node, it is fitted into the composite broadcast pattern. When a statelet family collapses due to a match or reverse linking, the composite broadcast pattern is reviewed, and extended for one or more statelet families which did not previously receive a full broadcast pattern at the node.

Reverse linking has the effect of dissolving a statelet family's mesh except for the subset of links traversing those nodes on the path of the reverse linking process. If no precursor from that family exists when the complemented statelet reaches a node, reverse linking is stopped. As long as reverse linking continues, a statelet family's mesh continues to collapse by suspending all transmit statelets from the given family except for the statelets on the newly created path itself. Suspension of these statelets causes them to disappear from the inputs of adjacent nodes, thereby collapsing the broadcast at those nodes as well, ultimately collapsing the entire mesh of any statelet family which successfully completes reverse linking onto a single path.

At the network level the effect is self-organizing contention amongst all statelet families to expand their mesh with the goal of completing a match and collapsing that mesh onto a path described by the trajectory of the reverse linking process. Freed links are incorporated into remaining meshes. If these meshes succeed in completing a match, they too collapse onto a path and make links available for other families and so on. All this occurs simultaneously, in parallel across the network and is limited only by node processing times, and link transmission and propagation delays.

This process is potentially very rapid because it proceeds asynchronously and exploits all parallel manners of completing a match at once. Significant parallelism is being exploited in both the spatial depth and breadth in the plane of the network; depth referring to the number of spares on each span, and breadth meaning the geographical diversity of the network topology.

Thus restoration proceeds self-paced until all statelet families have succeeded and collapsed onto a path, or halts when no remaining statelet families can succeed in expanding their mesh. In the latter case, a node suspends any remaining meshes by cancelling those unconfirmed statelets which it originated. Furthermore, at any time that 100% restoration is achieved during the dynamic restoration process, a source suspends any unneeded surplus statelets, collapsing unneeded meshes without waiting for a timeout. In either case, in the final state, only bidirectional complementary statelet pairs persist. Each pair runs the length of one continuous non-branching non-looping restoration path that condensed out of the mesh of one statelet family. The result is a mechanism, which results will show, produces link-disjoint loop-free pathsets that are very close to the multicommodity max-flow ideal obtainable otherwise only by centralized IP computation. Moreover, this is achieved by autonomous, database-free, self-organizing interaction between nodes.

# Chapter 9. OPRA Implementation Details

Chapter 9 presents the procedures required to effectively test OPRA using discrete event simulations. The nodal logic presented in this chapter supplement OPRA's core procedures presented in the previous chapter. The procedures presented in this chapter are needed to exploit the implementation technique presented in the following chapter.

## 9.1 Preventing Looping Event Sequences

OPRA usually allows a forward flooding statelet to be overwritten by a statelet from another family. Whenever a new forward flooding statelet with a lower interference number arrives at a node where all outgoing links contain non-null forward flooding statelets from other families, the new statelet is usually allowed to supplant another statelet which has not been complemented and has a larger interference number, as shown in Figure 9.1.

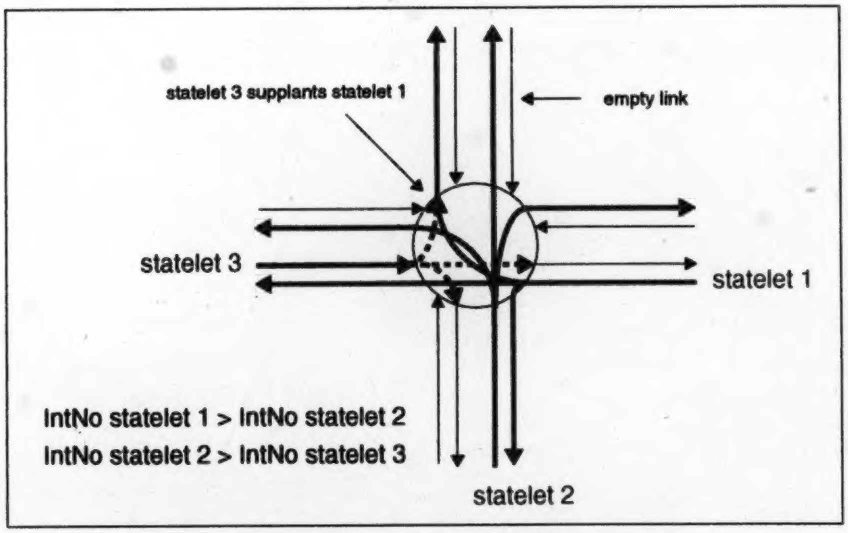

statelet 3 supplants statelet 1
empty link
statelet 3
statelet 1
IntNo statelet 1 > IntNo statelet 2
IntNo statelet 2 > IntNo statelet 3
statelet 2

Figure 9.1. Competition between statelets at a node

Because discrete event simulations advance in small finite time steps, a situation in which the statelets from two or more competing families continuously supplant one another my arise. As shown in Figure 9.2, in a discrete event simulation it is possible that a network will enter a cycle in which multiple nodes recognize a match, 9.2 (a), initiate reverse linking, 9.2 (b), stop reverse linking due to the disappearance of the forward flooding statelets, 9.2 (c), re-transmit new forward flooding statelets, 9.2 (d), and again recognize a match opportunity.

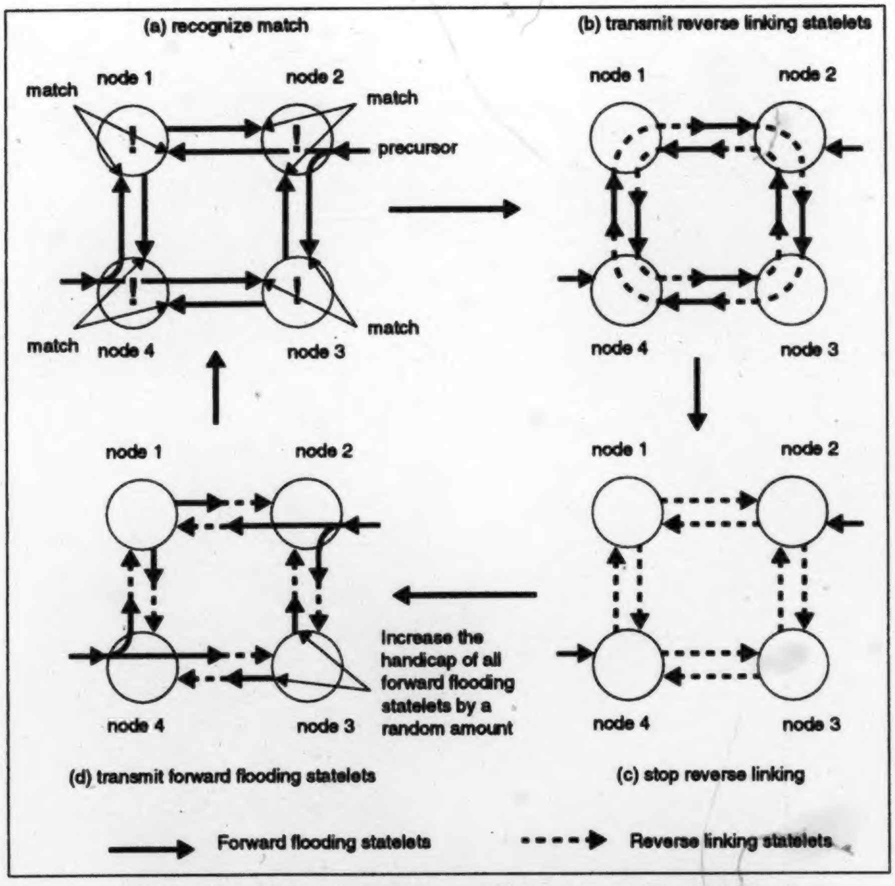

(a) recognize match
(b) transmit reverse linking statelets
node 1
node 2
node 3
node 4
match
precursor
!
Increase the handicap of all forward flooding statelets by a random amount
(d) transmit forward flooding statelets
(c) stop reverse linking
Forward flooding statelets
Reverse linking statelets

Figure 9.2. Using the handicap indicator to avoid oscillations

In reality this effect would likely resolve itself after one iteration, if it happened at all, because the event sequences to create a loop as shown in Figure 9.2 are unlikely to repeat themselves with the random timing variations present in the real world.

A key characteristic of OPRA is the supplanting of one statelet by another, and simulating this process is necessary when testing OPRA's ability to restore a failure in a transport network. In any discrete event simulation of OPRA, the supplanting of one statelet by another must be modelled, and a sequence of looping events as shown in Figure 9.2 resolved when it arises. To prevent such loops, a handicap is associated with every statelet at a node.

In the discrete event simulations of OPRA, the handicap of a statelet is entered into a separate information field along side those shown in Figure 7.1. The value of a statelet's handicap is propagated throughout a family's broadcast mesh and accumulates whenever it is increased by a node, similar to the way a statelet's repeat count accumulates.

When an outgoing statelet is overwritten, the value of its precursor's handicap is increased such that it is more difficult to supplant that outgoing statelet in subsequent competitions for broadcast. Before OPRA allows an incoming statelet with a lower interference number to supplant a statelet with a higher interference number as part of:

a) the forward flooding process, or
b) the *beginning* of the reverse linking process,
as shown in Figure 9.2,

the handicap of the incoming and outgoing statelet are compared. During forward flooding, an incoming statelet is allowed to overwrite an outgoing statelet only if the handicap of the incoming statelet multiplied by a weighting factor, named PRECURSOR-WEIGHTING, is larger than the handicap of the outgoing statelet. Similarly, reverse linking is only *initiated* if the handicap of the incoming matched statelets, after being multiplied by the constant MATCH-WEIGHTING, is greater than the handicap of any outgoing statelets from other relations being transmitted on one or both of the matched ports, as shown in Figure 9.2. Once reverse linking is initiated, it is allowed to continue as long as the forward flooding statelet with which a complement statelet is matched persists, regardless of the handicap of any outgoing statelet occupying the outgoing link on a matched port.

In a sense the handicap indicator assigns a higher priority to statelets which have been supplanted numerous times. To insure a statelet's handicap does not undo the desirable effects of competition based on the interference heuristic, it is increased by a very small amount, determined experimentally, so that the simple flux of events at a node don't lead to the growth of a broadcast mesh which violates the interference principle. Based on tests performed, a PRECURSOR-WEIGHTING and MATCH-WEIGHTING of 5 effectively eliminate any oscillations, i.e. non-self-settling interactions, without degrading OPRA's performance, when the handicap of a statelet is initially assigned a value, as well as increased by a random value, uniformly distributed between 0.1 and 1.

By slowly increasing the handicap indicator of all forward flooding statelets transmitted by a random amount, eventually a point is reached where some, but not all, of the nodes in a looping event sequence sustain a match or the broadcast pattern from one precursor. For example, in Figure 9.2, node 1 will only recognize a match between the red statelets received on spans 1-4 and 1-2 if the handicap indicator of the received statelets after being multiplied by MATCH-WEIGHTING are larger than each of the handicaps of the blue statelets sourced by node 1. If this node is the source of the statelet to be overwritten by the reverse linking statelet (e.g. nodes 1 and 3 in Figure 9.2), the upper and lower limits of the handicap assigned to a new forward flooding statelet are increased by a random variable uniformly distributed between 0 and 0.1. If the statelet to be overwritten is the branch of a precursor (e.g. the red statelets broadcasted by nodes 2 and 4 in Figure 9.2), the handicap of the precursor is increased by a random variable uniformly distributed between 0 and 0.1.

The constants MATCH-WEIGHTING and PRECURSOR-WEIGHTING control how quickly oscillations are stopped. Larger values will allow OPRA to oscillate for a longer period, while smaller values resolve the process very quickly. In the extreme, a value of zero for MATCH-WEIGHTING would prevent a match from supplanting any forward flooding statelets, and an infinite value for MATCH-WEIGHTING would always allow a match to supplant a forward flooding statelet. Similarly, a value of zero for PRECURSOR-WEIGHTING would prevent a forward flooding statelet from supplanting another forward flooding statelet, and an infinite value for PRECURSOR-WEIGHTING would always allow one forward flooding statelet to supplant another.

In conclusion, replacing and supplanting statelets is an integral part of OPRA; these actions occur frequently while evolving the composite restoration pathset. Oscillations of the nature shown in Figure 9.2 are very rare, and it was found that a relatively large value for MATCH-WEIGHTING and PRECURSOR-WEIGHTING compared to the value of a statelet's handicap quickly stopped any oscillations when they arose without degrading OPRA's ability to find a near optimal restoration pathset.

## 9.2 Correct Sequencing of Port Updates

When processing a single interrupt, it may be required to transmit two or more statelets on a single span. When OPRA is implemented such that not all new statelets on a span are presented to the DRA at once, unlike the implementation presented in the following chapter, the sequence in which those ports are latched to their respective links should follow the same sequence OPRA followed in altering the contents of the transmit statelet registers. For example, when the branch from a matched precursor on span x resides on span y, and the precursor to which the precursor on span x is matched also resides on span y, but not the same port, as shown in Figure 9.3, and new statelets are processed immediately without waiting for all of the links on a span to be updated, the branch on span y should be eliminated before transmitting the complemented statelet. Latching all registers onto links in the same order in which they were updated internally in OPRA's execution sequence ensures that OPRA processes statelets within one span in the proper sequence, regardless of how OPRA is implemented.

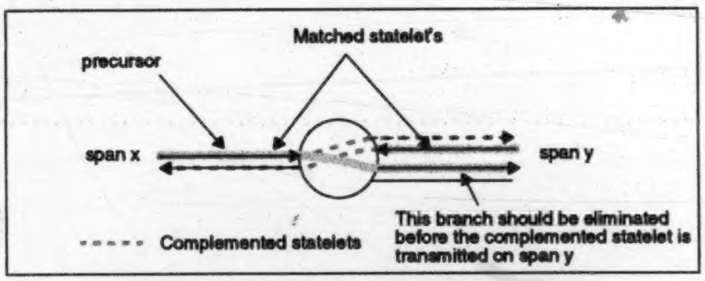

Matched statelet's
precursor
span x
span y
This branch should be eliminated before the complemented statelet is transmitted on span y
----- Complemented statelets

Figure 9.3. Correct sequencing of port updates

## 9.3 Re-Initiating Reverse Linking

Depending on the implementation of OPRA, the TxReg of a port which is part of a reverse linking process, i.e. *activated* = true, may have to be cancelled and retransmitted after a forward flooding statelet overwrites an existing precursor from its own family in order to continue reverse linking. After the start of a restoration event, if OPRA is implemented such that a mechanism external to OPRA is responsible for informing the DRA of a new statelet arrival, like the polling mechanism explained in the following chapter, the arrival of a statelet, qualified as best, which overwrites an existing precursor from its own family on a activated port, requires retransmitting the entry in that port's TxReg register, as explained below.

Two ports become activated when a match is found, as shown in Figure 9.4 (a). Reverse linking continues at a tandem node, node 2 in Figure 9.4, as long as the forward propagating statelet matched to the complemented statelet persists at that node.

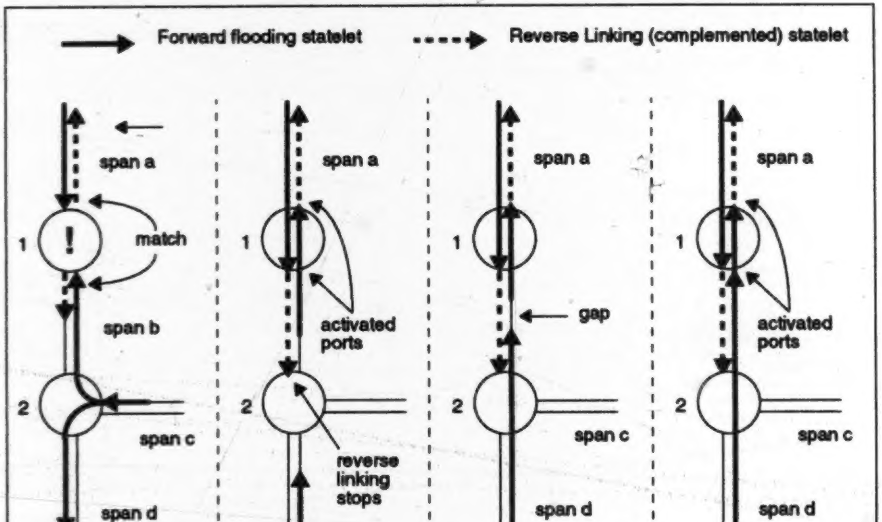

Forward flooding statelet
Reverse Linking (complemented) statelet
span a
1
!
match
span b
2
span c
span d
(a)
span a
1
activated ports
2
reverse linking stops
(b)
span a
1
gap
2
span c
span d
(c)
span a
1
activated ports
2
span c
span d
(d)

Figure 9.4. Overwriting a precursor from the same statelet family on a activated port

As shown in Figure 9.4 (b), if the precursor on span c disappears at node 2 and the forward flooding statelet to node 1 on span b is cancelled before the complemented statelet on span b arrives at node 2, reverse linking stops. It is possible that a new forward flooding statelet from the same family as the vanished precursor on span c arrives on span d at node 2 shortly afterwards, and becomes that family's new precursor as shown in Figure 9.4 (c). Unless the DRA implementation is somehow able to inform OPRA of the existence of the reverse linking statelet at node 2 without the RxRS flag being set, the complemented statelet on span b at node 2 will not continue reverse linking and follow the path traced by the new precursor on span d until node 1 cancels and retransmits that same complemented statelet.

Therefore, when

1) the time interval between the disappearance and reappearance of a statelet from the same family is very short, as shown by the "gap" in Figure 9.4 (c), and
2) OPRA only processes those ports whose RxRS flag is raised at discrete time intervals, as explained in the following chapter,

node 1 may not process the first disappearance event, because possibly another statelet is being processed at that node, and determine that a statelet overwrote a precursor from the same family on a activated port when processing the second reappearance event. Unless node 1 cancels and retransmits the complemented statelet on span d, the reverse linking event halted at node 2 is never restarted, as shown in Figure 9.4 (d). Restarting the reverse linking process is desirable because it may lead to the formation of a restoration path. In the discrete event simulations performed, given the situation depicted in Figure 9.4, OPRA cancelled and retransmitted the entries in the TxReg register of node 1 on span b.

## 9.4 Optimizing the Forward Flooding Process

To reduce the number of times a statelet is supplanted when a match is completed, and thereby possibly reduce the restoration time, it is advantageous to choose different spares on a connecting span when both nodes terminating that span are sourcing forward flooding statelets from different relations. As shown in Figure 9.5 (a), when nodes 1, 2, and 3, transmit forward flooding statelets on links 4, 5, and 6, a reverse linking event destined to fail is initiated in nodes 2 and 3. Reverse linking is stopped at node 2 when the blue forward flooding statelet is overwritten by the red complemented statelet, and at node 3 when the red forward flooding statelet is overwritten by the blue complemented statelet. However, when nodes 1, 2, and 3, transmit forward flooding statelets as shown in Figure 9.5 (b), reverse linking is completed and two restoration paths are found. As shown in Figure 9.5 (b), OPRA is implemented such that different spares on a connecting span are used when initiating forward flooding statelets at a source whenever possible so as to minimize initiating reverse linking processes destined to fail.

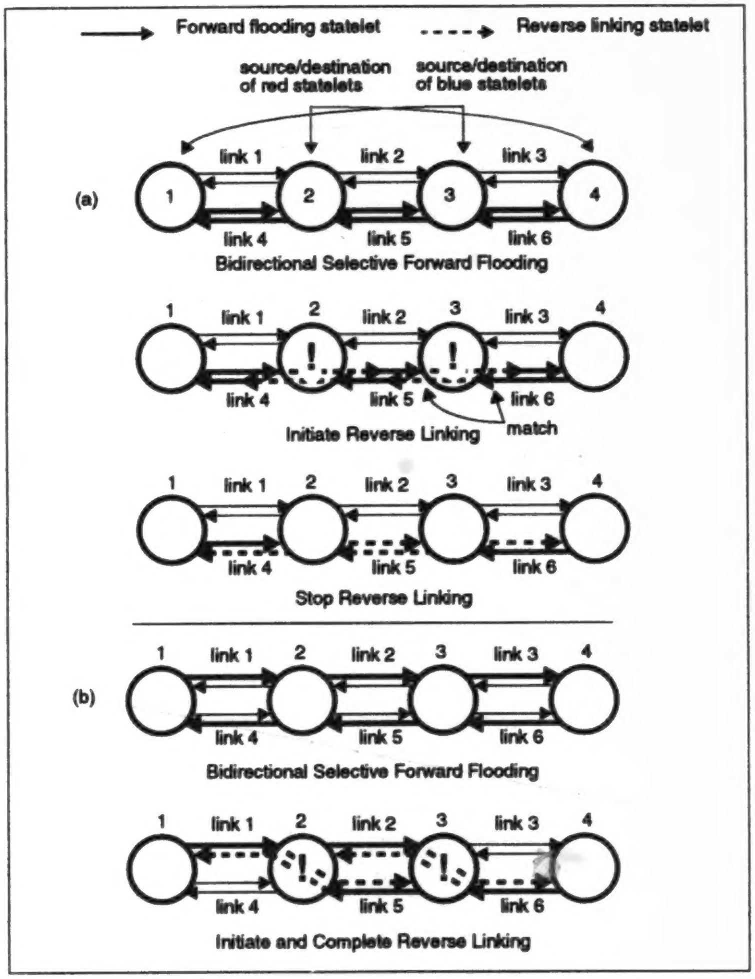

Forward flooding statelet
Reverse linking statelet
source/destination of red statelets
source/destination of blue statelets
(a)
link 1
link 2
link 3
link 4
link 5
link 6
1
2
3
4
Bidirectional Selective Forward Flooding
match
Initiate Reverse Linking
Stop Reverse Linking
(b)
Bidirectional Selective Forward Flooding
Initiate and Complete Reverse Linking

Figure 9.5. Initiating forward flooding statelets on different spares on a connecting span

# Chapter 10. Implementation of a Simulation Test-Bed for OPRA

This chapter describes the testbed in which OPRA was implemented for experimental characterization. Subsequent chapters report various results obtained with this experimental implementation of OPRA.

The complexity of the asynchronous, event-driven, mutual interactions, between nodes executing OPRA is a complex system level problem. By its nature, OPRA is an asynchronous, massively parallel, multi-dimensional transient process. It engenders many different types of constructive interactions between statelets; the state space of the node and link combinations is large and changes with time. For these reasons an experimental validation and research development method has been necessary, as opposed to seeking theoretical proofs of performance. The main technique used in this work is computer emulation of OPRA running concurrently at every node of representative networks. In this approach an executable single-processor implementation of the intended protocol is actually constructed and instances of it executed concurrently in every node of the various study networks using a concurrent programming, discrete event simulation environment. The collective behaviour of the network as a large scale system is then observed.

## 10.1 Modelling Environment

There are two completely separate and replaceable parts to the experimental system which was used to test and develop OPRA:

1. the module that embodies OPRA, and
2. the testbed.

The testbed builds an environment within which OPRA operates as if identical instances of it resided in the DCS machines of a real network. The testbed converts one physical instance of OPRA into many virtual (concurrent) instances and facilitates their distributed asynchronous interaction according to a defined network description. This experimental system reveals how small changes in the statelet processing rules impact the large scale behaviour of the network. The research was mechanized this way to investigate complex statelet interactions in large networks, and refine the logic and processing rules that constitute OPRA.

### 10.1.1 Modelling OPRA

Given that a single specification of OPRA is needed which can be executed by many nodes simultaneously, OPRA was written as a set of rules using finite state machine (FSM) techniques. The testbed can therefore execute one instance of OPRA for one node, suspend it, and use the same specifications to advance the processing for another node, and so on.

Using FSM techniques in the specification of OPRA allows all behaviours to be described in terms of actions in response to external events. FSM methods are often used in telecommunications software design because of their natural pacing of responses to external events with inherent task suspension when no events require processing. Several benefits arise from using FSM techniques to specify OPRA:

1. It allows very structured thinking about problems related to the algorithm in localized terms such as "if the current state is x, and event y occurs, then what is the action to take?" Every action in response to an event is itself a conventional algorithmic segment designed to realize the intended function. The isolation from the overall problem complexity using this structuring is helpful and desirable as a discipline in the development of a software system for the massively parallel interactions that distributed mesh restoration represents.

2. Considerable confidence is obtained about the practicality of OPRA in real crossconnect machines because the code is written as it would be to run within the operating system of a DCS machine in a real network. Furthermore, the interface between the testbed and the protocol specification can be made to appear as it would in a real DCS machine that passes events through that interface to and from the other nodes of the network, as well as applying appropriate insertion, propagation, and execution time delays.
3. The testbed actually executes a specification of the algorithm. Thus, when the desired network-level behaviour is obtained, one can print out the algorithm, and obtain an executable C language specification of the DRA.

### 10.1.2 The Testbed

The testbed executes multiple instances of OPRA concurrently in a manner similar to the way a multi-tasking operating systems executes several instances of a given application program. The testbed builds an environment in which each instance of OPRA sees the environment of a single node interacting directly with the network in which it is embedded. Any statelet applied to a link is delivered after appropriate delays to the respective port at the node that is connected to the receiving end of that link, according to the network description file. Unlike the module used to implement OPRA, the testbed knows the network's design and converts any changes one node makes to the transmit links at its site into future events occurring at other nodes. The testbed simulates the interface to the DCS and passes events through it to and from other nodes in the network using appropriate transmission, propagation, and execution time delays. Within the testbed, OPRA's view of the world is the same in all relevant aspects as it would be in a real DCS machine following the specifications presented in chapter 8.

## 10.2 Testbed Description

The testbed was built using OPNET [52]. OPNET is a comprehensive development environment supporting the modelling and performance evaluation of communication networks and distributed systems involving concurrent processes. System behaviour and performance are analysed by discrete event simulation.

OPNET is used in this work to provide the basic virtual-time scheduling utilities for concurrent asynchronous execution of OPRA with realistic propagation, transmission and processing delays. Link propagation and other delays are calculated and incorporated in the scheduling of future events. An event originating at time *now* and place *here* is scheduled so as to have its effect as an event occurring somewhere else in the network determined by the network file at time (*now + delay*), where *delay* can be computed or assigned by the test bed. OPNET manages the advance of virtual time in a manner which allows different nodes to execute OPRA at overlapping moments in virtual time. All events are scheduled on the time line, and nodes running OPRA task instances are only invoked at those points in virtual time when an event occurs for them.

When active, a node executes OPRA according to its last state stored by OPNET and the DCS port records visible to it. This may result in the creation of new statelets or modification of currently transmitted statelets. When the node suspends execution of OPRA after one event-action-next state cycle as described in chapter 8, any future events which have been caused by the execution of this node are put in a queue of events scheduled for some future instant of virtual time. Then OPNET processes the next event scheduled for the present instant in virtual time. When all events scheduled for this instant in virtual time have been processed, the virtual clock is incremented to the next instant in virtual time at which one or more events are scheduled. This series of steps continues until no future events are scheduled.

In addition to OPNET's virtual-time scheduling utility, the testbed uses OPNET objects, each with a configurable set of attributes. Collectively, these objects form a hierarchial OPNET model, and the Network, Node, and Process modelling domains of OPNET span all the hierarchial levels of the model. The structure of the network testbed is described in terms of OPNET's Network, Node, and Process domains below.

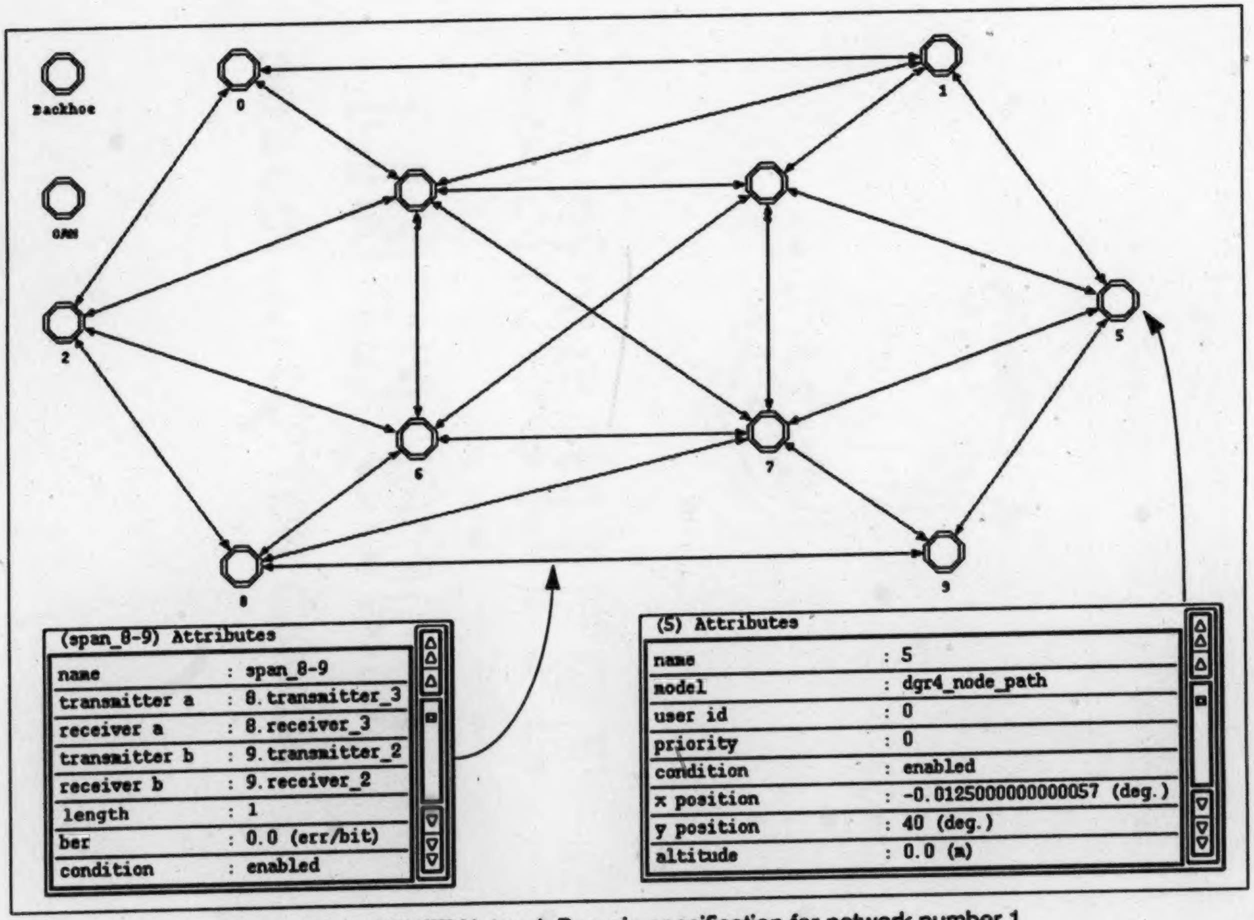
### 10.2.1 OPNET's Network Domain

As part of the testbed, OPNET's Network Domain is used to define the topology of a transport network, which includes specifying the location of nodes and the working and spare links between nodes. Figure 10.1 shows the OPNET Network Domain specification for network number 1 presented in chapter 4. In the Network Domain the following attributes are defined:

1. the receiver and transmitter terminating a span,
2. the length of a span,
3. the number of spare and working links in a span terminated at a node, and
4. the capability of a node as explained below.

The specific capabilities of a node are defined by designating its model. For example, in Figure 10.1 the model *dgr4_node_path* is designated for node 5. The specifics of *dgr4_node_path*, which stands for a degree 4 node model capable of performing path restoration, are presented in the following section. Attributes 1 through 4 can be specified using OPNET's Network Domain graphical editor, however, to expedite testing OPRA in many networks, these values were set with a program written at TRLabs that uses OPNET's External Module Access (EMA) capability.

Two of the "nodes" shown in Figure 10.1 are not part of the network's topology. Instead of DCS machines, the "nodes" named *Backhoe* and *OAM* represent devices needed to cut one or more spans in a network and to record information on completed restoration paths respectively. Details of these nodes will be presented towards the end of this chapter.



(span_8-9) Attributes

name : span_8-9
transmitter a : 8.transmitter_3
receiver a : 8.receiver_3
transmitter b : 9.transmitter_2
receiver b : 9.receiver_2
length : 1
ber : 0.0 (err/bit)
condition : enabled

(5) Attributes

name : 5
model : dgr4_node_path
user id : 0
priority : 0
condition : enabled
x position : -0.0125000000000057 (deg.)
y position : 40 (deg.)
altitude : 0.0 (m)

Figure 10.1. OPNET Network Domain specification for network number 1

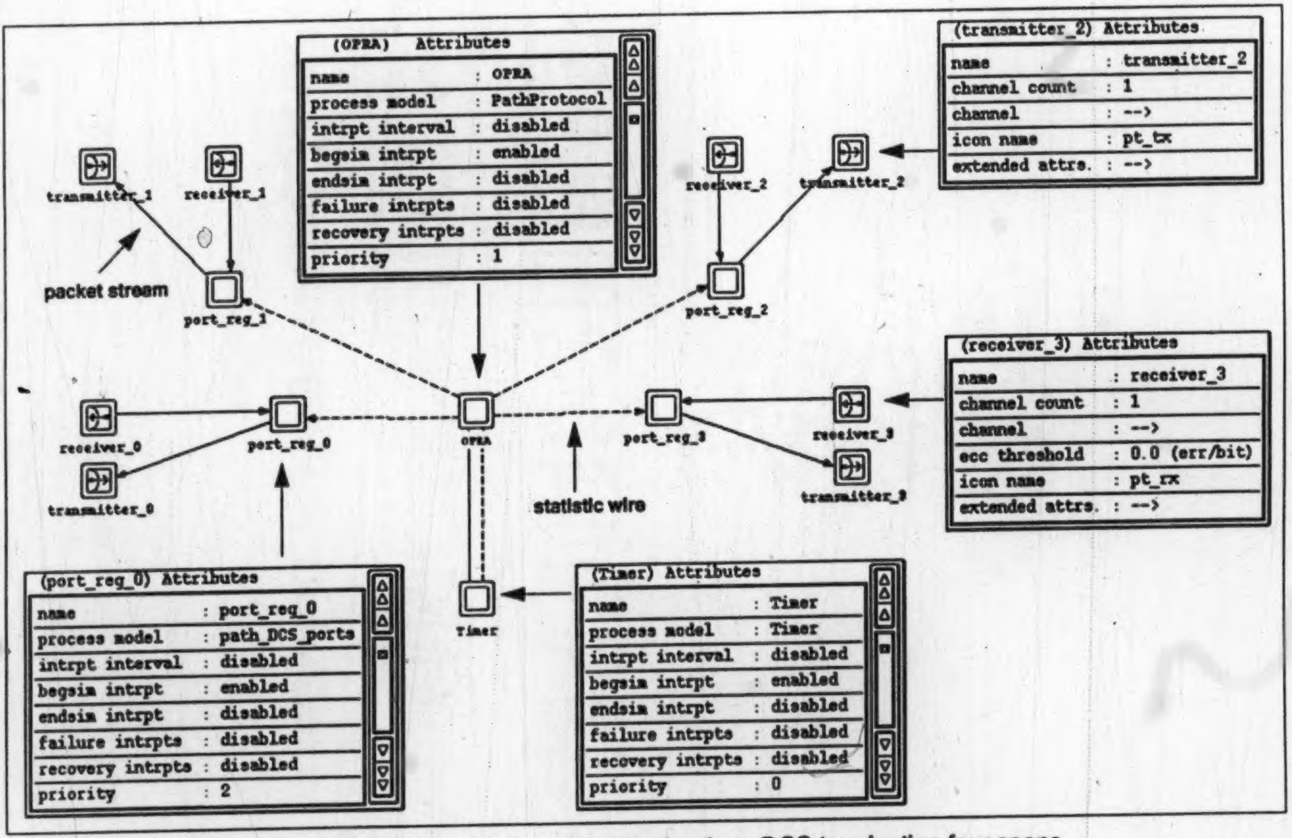
### 10.2.2 OPNET's Node Domain

OPNET's Node Domain provides for the modelling of communication devices that can be deployed and interconnected at the network level. In OPNET terms these devices are called nodes, and in the context of mesh restoration, they correspond to DCSs.

The node model for a DCS which terminates four spans, i.e. node model *dgr4_node_path*, is shown in Figure 10.2. Node models are expressed in terms of smaller building blocks called modules. In Figure 10.2, *Timer, transmitter_1, receiver_1, port_reg_1* etc. are all modules. All of the receiver and transmitter module's capabilities are substantially predefined and are configured through a set of built-in parameters as shown. The receiver and transmitter modules attach a node (DCS) to communication links in the Network Domain. All of the other modules are highly programmable, their behaviour being prescribed by an assigned *process model*. OPRA is the process model for the module *OPRA*. The process models inside the *Timer* and *port_reg* modules are presented later in this chapter.

A node model may consist of any number of modules of different types. For a DCS terminating 3 spans, instead of four as shown in Figure 10.2, one *receiver, transmitter, and port_reg* module triplet would be removed, and for a DCS terminating five spans, one *receiver, transmitter, and port_reg* module triplet added. Following this technique, the base node model shown in Figure 10.2 can be modified to represent any degree DCS. Only one *receiver, transmitter,* and *port_reg* triplet is required per span because a network's link structure is resolved inside the *port_reg* module. Given that real networks may have hundreds of links per span, each statelet transmitted or received at the *port_reg* module is tagged with a number identifying the logical link with which it is associated.

As shown in Figure 10.2, two types of connection are provided to support interaction between modules. These are called packet streams (solid lines) and statistic wires (dashed lines). Statistic wires convey control information between modules. Packet streams allow formatted messages called packets in OPNET to be conveyed from one module to another. This does not mean packet message instead of state based interaction is being modelled, but that OPNET's paradigm of a packet is being used to convey a single statelet change event.



transmitter_1
receiver_1
packet stream
port_reg_1
receiver_0
transmitter_0
port_reg_0
OPRA
Timer
statistic wire
receiver_2
transmitter_2
port_reg_2
port_reg_3
receiver_3
transmitter_3

(OPRA) Attributes
name : OPRA
process model : PathProtocol
intrpt interval : disabled
begsim intrpt : enabled
endsim intrpt : disabled
failure intrpts : disabled
recovery intrpts : disabled
priority : 1

(transmitter_2) Attributes
name : transmitter_2
channel count : 1
channel : -->
icon name : pt_tx
extended attrs. : -->

(receiver_3) Attributes
name : receiver_3
channel count : 1
channel : -->
ecc threshold : 0.0 (err/bit)
icon name : pt_rx
extended attrs. : -->

(port_reg_0) Attributes
name : port_reg_0
process model : path_DCS_ports
intrpt interval : disabled
begsim intrpt : enabled
endsim intrpt : disabled
failure intrpts : disabled
recovery intrpts : disabled
priority : 2

(Timer) Attributes
name : Timer
process model : Timer
intrpt interval : disabled
begsim intrpt : enabled
endsim intrpt : disabled
failure intrpts : disabled
recovery intrpts : disabled
priority : 0

Figure 10.2. OPNET Node Domain specification for a DCS terminating four spans

### 10.2.3 OPNET's Process Domain

The tasks performed by *port_reg*, *Timer*, and *OPRA* of the Node Domain are specified in the Process Domain using process models. Each process model is defined using a combination of state transition diagrams, a library of high level OPNET kernel procedures, and the general facilities of the C programming language. A process model's state transition diagram defines a set of primary states that the process can enter, and for each state, the conditions that would cause the process to move to another state. The condition for a particular change in state and the associated destination state are called a *transition*. Each state defines a set of actions which must be performed upon entering or exiting that state. The actions can be extremely general in nature because they are expressible as C language statements.

One process model, OPRA, has already been explained in detail. OPRA resides inside the *OPRA* module and is assigned the name *PathProtocol* within the Process Domain. The C-code and OPNET kernel needed to implement OPRA are given in [40]. The process model inside the *port_reg* and *Timer* modules shown in Figure 10.2. are explained next.

## 10.3 Port Card Process Model

The functions associated with the port card of a DCS performing restoration are defined in the *path_DCS_ports* process model. The *path_DCS_ports* process model is responsible for receiving and transmitting statelet change events (as OPNET packets), receiving alarms, and calculating the transmission and propagation delay of the span connected to the port card. This process model implements the RxReg, TxReg, and PSReg registers in accordance with the SONET DCS architecture presented in chapter 2. The state transition diagram of *path_DCS_ports* is shown in Figure 10.3. In Figure 10.3, the meaning of, and differences between, light and dark coloured states, and solid and dashed transitions, are the same as explained in chapter 8 for Figure 8.1.

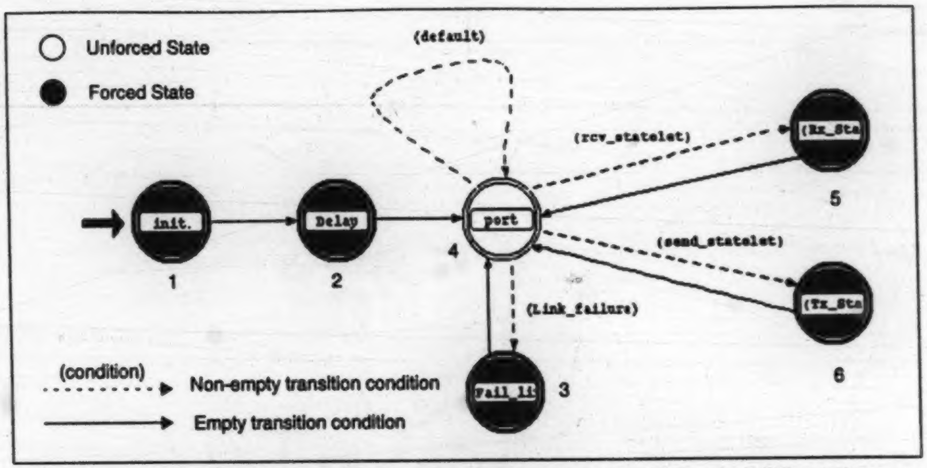

Unforced State
Forced State
(default)
(rcv_statelet)
(Rx_Sta
5
init.
Delay
port
4
1
2
(send_statelet)
(Tx_Sta
(Link_failure)
6
(condition)
Non-empty transition condition
Empty transition condition
Fail_li
3

Figure 10.3. State transition diagram for process model *path_DCS_ports*

State 1, *init*, shown in Figure 10.3 is responsible for initializing all of the TxReg, RxReg, and PSReg registers connected to the span associated with this process model.

State 2, *Delay*, is responsible for calculating the propagation and transmission delay of any statelet transmitted on the span associated with this port. Assuming that one of the unused bytes in the Line Overhead (LOH) of a SONET transport signal is used for statelet transmission, and given that the size of a statelet is 80 bits, the serial insertion delay for all spans in a network was fixed at 1.25 msec (80 bits / 64 000 bits/sec). The propagation delay was calculated assuming fibre optic transmission systems would be used to transmit a statelet: *propagation delay* = (*span length in km*) / (0.7 x $3\times10^5$ km/sec).

State 4 is the idle state in which the port process model will remain and return after either receiving an alarm or statelet, or transmitting a statelet. When an interrupt is received, the idle state (i.e. state 4) determines which of these events is taking place.

When a port receives an alarm, state 3, *Fail_Link*, is entered. *Fail_Link* is responsible for setting the *LineAlarm*, *PathAlarm*, *Trace*, and *AIS* fields in the PSReg of the failed port. The *RxRS* flag of the failed port is also raised to indicate an alarm needs processing by OPRA. If the alarm received is the first, the polling mechanism used to service all ports at a DCS, described in the following section, is started.

When a complete statelet is received at a port, and that statelet is different from the one stored in the RxReg register, state 5, *Rx_Statelet_Reg*, is entered, *Rx_Statelet_Reg* latches the new statelet into the receive register and informs OPRA of the change using the *RxRS* field. As in the *Fail_Link* state, if the statelet received is the first, the polling mechanism used to service all ports at a DCS is started, if not already active.

When a statelet needs to be transmitted, state 6, *Tx_Statelet_Reg*, is entered. *Tx_Statelet_Reg* latches the new statelet onto the link terminated at that port after waiting a period of time equal to the sum of the propagation and transmission delay.

The C-code and OPNET kernel associated with the Port Card Process Model are given in [39].

## 10.4 Polling Mechanism Process Model

It is possible that multiple restoration statelets and/or alarms arrive at a node while OPRA is responding to an interrupt. Furthermore, OPRA might ignore a statelet while processing an interrupt if the statelet changes several times on a single link. Therefore, OPRA is designed to handle multiple interrupts simultaneously and function correctly even after ignoring a number of statelet changes. The tasks OPRA performs depend only on the states of the ports at a node at the instant a transition is initiated. This processing is implemented by sequentially polling ports that have received a new statelet or an alarm since the last poll (see Figure 10.4).

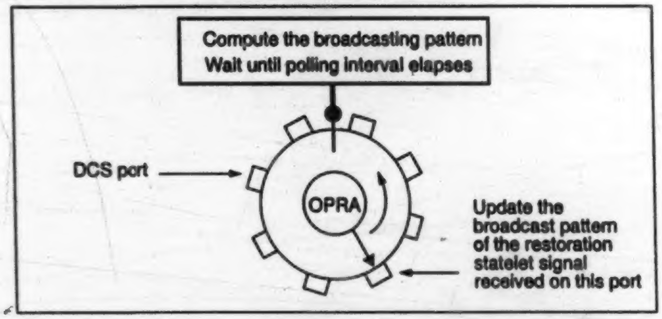

Compute the broadcasting pattern
Wait until polling interval elapses
DCS port
OPRA
Update the broadcast pattern of the restoration statelet signal received on this port

Figure 10.4. Model of the polling mechanism used by OPRA

Using a polling mechanism to sequentially cycle through all links terminated on a DCS, acknowledges interrupts in an orderly fashion and controls the number of interrupts generated while restoring a failure. Servicing interrupts at a node usually requires OPRA to transmit one or more statelets. Therefore, increasing the time between polls will increase the number of statelet changes ignored and decrease the number of statelets generated. A polling interval of zero would have OPRA poll all ports continuously, checking each port after a delay determined by the time required to:

1) process all ports whose link state has changed,
2) re-compute the broadcast pattern at a node, and
3) poll each port.

Though the broadcast pattern associated with an individual statelet is updated every time a port is polled, the broadcast pattern at a node is computed only after all ports have been serviced. Updating the broadcast pattern of a node may involve cancelling restoration statelets on other links by pruning the branches of a precursor, reverse linking a complemented restoration statelet, moving the root of a precursor, and/or transmitting a loop back test signal; however, it does not entail broadcasting new forward flooding restoration statelets. Only computing the broadcast pattern at a node can result in transmitting new forward flooding restoration statelets. This computation is performed at the end of a polling cycle because computing the broadcast pattern requires that the most recent interference numbers of all restoration statelets received at a node be considered.

The functions associated with the polling mechanism which OPRA uses to service ports that receive a new statelet or alarm are defined in the *Timer* process model shown in Figure 10.5. State 1, *init*, is responsible for initializing the process model. State 2, *Trigger*, is an empty state in which the Timer process model remains until activated by either state 3, *Fail_Link*, or state 5, *Rx_Statelet_Reg*, of the *path_DCS_ports* process model.

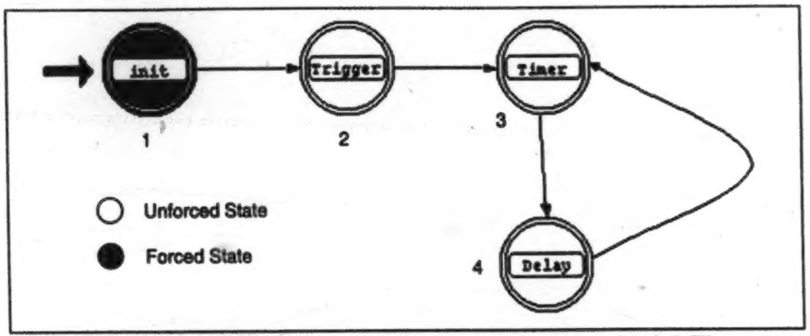


Figure 10.5. State transition diagram for process model *Timer*

The polling mechanisms of source/destination nodes are activated by alarms in the case of an instantaneous span cut, and by either alarms or forward flooding statelets in the case of a staggered alarm scenario. The polling mechanism of a tandem node is activated by a forward flooding statelet in either case. Because some nodes are activated by alarms, while others are activated by a forward flooding statelet, in both cases the polling mechanisms of all nodes are rarely synchronized.

Upon activating the polling mechanism *Timer* is entered. If the time allotted to restore a failure has not expired, *Timer* schedules another polling cycle in which all ports whose *RxRS* flag is true are processed. The idle period between polling cycles is set by the constant DELTA_T. In addition, upon entering *Timer*, all future external interrupts are prevented from interrupting the polling mechanism so that once the polling mechanism is activated it polls all ports at a constant rate until the time allotted to complete a restoration event has expired.

After waiting the prescribed time between polling cycles, DELTA_T, the location of all ports whose *RxRS* flag is set to true are forwarded to OPRA at once in the form of a list for processing. While OPRA processes those ports whose *RxRS* flag was raised during a polling cycle, new alarms and statelets are prevented from overwriting the contents of the PSReg and RxReg registers. Given the SONET DCS architecture presented in chapter 2, this would require preventing the first receive register of all ports from latching to the second register. Holding the contents of the second register of all incoming links constant during a single polling cycle ensures OPRA maintains a static picture of all ports for the brief period it is servicing any ports whose *RxRS* flag was raised. Whether or not some link state changes are ignored during this time interval is irrelevant because only the current state of all links at a node at the instant OPRA is activated determines the trajectory which is traced through OPRA's state transition diagram, Figure 8.1.

Though OPRA is insensitive to the order in which ports are polled, it is beneficial to service all forward flooding statelets before servicing any reverse linking statelets. Updating the broadcast pattern of all forward flooding statelets before updating the broadcasting pattern of all complemented statelets ensures reverse linking statelets trace back the path identified by the most recent precursor. While tracing back the path identified by a precursor which is destined to be eliminated or replaced is a legitimate action dealt with by OPRA, it generates spurious statelet change events. Therefore, the polling mechanism employed by OPRA updates the broadcast pattern of all forward flooding statelets before updating the broadcast pattern of all reverse linking statelets.

After completing one polling cycle, the polling mechanism sends OPRA a signal informing it that it should compute the composite broadcast pattern at this node. As detailed in chapter 8, updating the broadcast pattern of an individual port may involve cancelling restoration statelets on other links, reverse linking a complemented statelet, moving the root of a precursor, and/or transmitting a loop back test signal; however, it does not entail broadcasting new forward flooding statelets. Only computing the composite broadcast pattern at a node involves the possibility of transmitting new forward flooding statelets.

A single list of all altered ports ending with an indication to re-compute this node's broadcast pattern is submitted by the polling mechanism to OPRA for processing. The processing delay associated with updating all ports on that list, as well as the time required to compute the composite broadcast pattern at this node, is accounted for in the polling mechanism's process model by waiting in state 4, *Delay,* until OPRA has processed all altered ports and re-computed a node's broadcast pattern.

Given that a static picture of all ports is maintained for the brief instant OPRA services any ports which had their *RxRS* flag raised, the polling mechanism effectively controls the pace at which OPRA processes statelets and alarms. As long as the polling interval, plus the processing delay, dominate the propagation and insertion delays combined, restoration time is determined mainly by node delays. Otherwise, restoration time is limited by the lengths of links and the bandwidth available to statelets. These effects are examined in more detail in the following chapters.

The time required to process a new statelet or alarm is set by the constant STATELET_PROCESSING_DELAY. In most of the network restorability tests performed, this was set at 1 msec. Given a DCS CPU clock rate of 200 MHz, and assuming a C-level instruction takes 5 clock cycles to complete, approximately 40 thousand lines of C-code may be executed in 1 msec. Processing one port requires in the worst case on the order of a few hundred lines of C-code; 1 msec is therefore a very conservative estimate of the time required to process an interrupt generated by either an alarm or a statelet. Furthermore, a processing delay of 1 msec is twice the value chosen previously [21].

An even larger processing delay was assumed for the time required to compute a node's composite broadcast pattern. In the worst case, computing the composite broadcast pattern at a node may require servicing all ports at a node. BROADCAST-DELAY was therefore set at a conservative value of 2 msec.

After a time equal to the processing delay of all ports whose *RxRS* flag was set in the last polling cycle plus the time required to compute a node's broadcast pattern, the *Delay* state shown in Figure 10.5 is exited.

The C-code and OPNET kernel associated with the Polling Mechanism Process Model are given in [39].

## 10.5 Initiating a Network Failure

The node named *Backhoe* in the OPNET Network Domain shown in Figure 10.1 is responsible for disabling all the spans impacted by a failure. Up to 6 different spans may be failed simultaneously, or randomly in time. Given the ability to fail 6 spans simultaneously, OPRA's ability to restore node failures, multiple span failures, and single span failures can be tested using the network emulator.

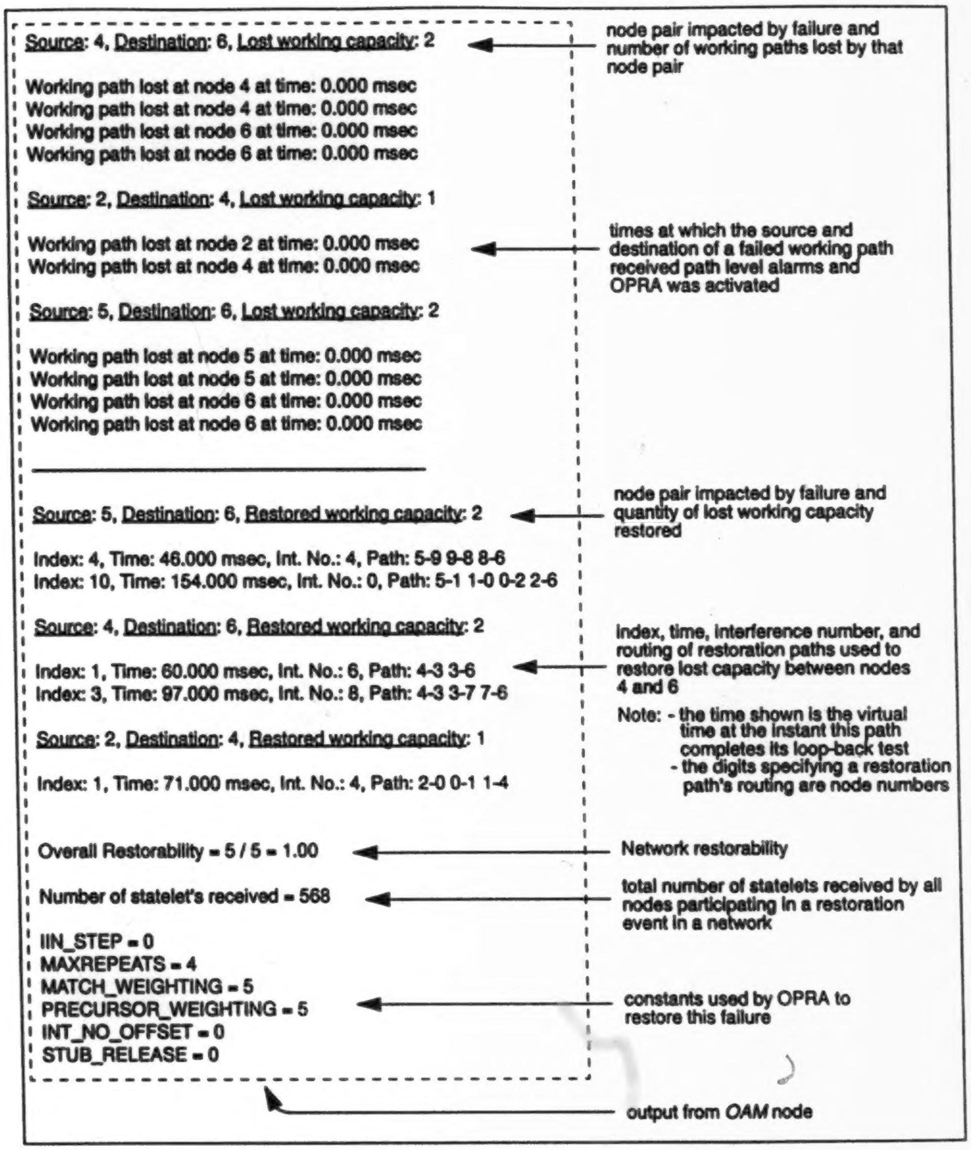
Failing a span requires determining what working paths traverse the failed span and setting the appropriate alarms on all ports traversed by the end-to-end working path. This function is performed by node *Backhoe*. The line alarm of all spare links on the failed span(s) are also set by node *Backhoe*.

The C-code and OPNET kernel associated with the *Backhoe* Process Model are given in [39].

## 10.6 Collecting Working and Restoration Path Data

After a restoration event is completed, the node *OAM* records information on the working paths severed and the restoration paths found by OPRA. Node *OAM* is analogous to a network's operation and control centre that would oversee realtime restoration. A sample of the output generated by the C code inside the *OAM* node is shown in Figure 10.6.

The C-code and OPNET kernel associated with the *OAM* Process Model are given in [39].



```
Source: 4, Destination: 6, Lost working capacity: 2

Working path lost at node 4 at time: 0.000 msec
Working path lost at node 4 at time: 0.000 msec
Working path lost at node 6 at time: 0.000 msec
Working path lost at node 6 at time: 0.000 msec

Source: 2, Destination: 4, Lost working capacity: 1

Working path lost at node 2 at time: 0.000 msec
Working path lost at node 4 at time: 0.000 msec

Source: 5, Destination: 6, Lost working capacity: 2

Working path lost at node 5 at time: 0.000 msec
Working path lost at node 5 at time: 0.000 msec
Working path lost at node 6 at time: 0.000 msec
Working path lost at node 6 at time: 0.000 msec

________________________________________

Source: 5, Destination: 6, Restored working capacity: 2

Index: 4, Time: 46.000 msec, Int. No.: 4, Path: 5-9 9-8 8-6
Index: 10, Time: 154.000 msec, Int. No.: 0, Path: 5-1 1-0 0-2 2-6

Source: 4, Destination: 6, Restored working capacity: 2

Index: 1, Time: 60.000 msec, Int. No.: 6, Path: 4-3 3-6
Index: 3, Time: 97.000 msec, Int. No.: 8, Path: 4-3 3-7 7-6

Source: 2, Destination: 4, Restored working capacity: 1

Index: 1, Time: 71.000 msec, Int. No.: 4, Path: 2-0 0-1 1-4


Overall Restorability = 5 / 5 = 1.00

Number of statelet's received = 568

IIN_STEP = 0
MAXREPEATS = 4
MATCH_WEIGHTING = 5
PRECURSOR_WEIGHTING = 5
INT_NO_OFFSET = 0
STUB_RELEASE = 0
```

node pair impacted by failure and number of working paths lost by that node pair

times at which the source and destination of a failed working path received path level alarms and OPRA was activated

node pair impacted by failure and quantity of lost working capacity restored

index, time, interference number, and routing of restoration paths used to restore lost capacity between nodes 4 and 6

Note: - the time shown is the virtual time at the instant this path completes its loop-back test
- the digits specifying a restoration path's routing are node numbers

Network restorability

total number of statelets received by all nodes participating in a restoration event in a network

constants used by OPRA to restore this failure

output from *OAM* node

Figure 10.6. Sample output generated by *OAM* node

## 10.7 Verifying OPRA's Operation

Given the complexity of OPRA and the transient asynchronous event-driven interactions associated with distributed restoration, OPRA's execution must be traced to verify it. In small test networks OPRA's functionality was verified by manually tracing the trajectories of all statelet families for entire restoration events. These tests ensured that OPRA found a near-optimal restoration pathset, did not violate the capacity constraints on any span, and functioned as explained previously.

For larger networks it was impossible to trace OPRA's execution by hand. Consequently numerous *assert* statements were built into the C-code specification of OPRA to define checkpoints at which one could verify its operation. Whenever OPRA entered one of the states shown in Figure 8.1, most conditions which had to be true (or false) while in that state were verified using assert statements. For example, in the *Tandem_Node* state (state 1) shown in Figure 8.1, OPRA verified that if a precursor existed previously at a port, the port was not empty.

As a final test of OPRA's operation and the consistency of the restoration pathset found with the constraints presented in chapter 3, animations of various restoration events were produced. Using movie-like animations of OPRA, it was possible to verify that:

1. restoration paths were consistent with a network's topology,
2. OPRA stopped executing only when all lost capacity had been restored or no other restoration paths were topologically feasible,
3. the total number of forward flooding, reverse linking, and confirmation statelets on a span never exceeded the available link capacity on that span,
4. forward flooding statelets did not trace looping paths,
5. reverse linking statelets followed a path paralleling the forward flooding statelet with which they were paired,

6. looping event sequences of the type described in chapter 9 never persisted,
7. only those nodes impacted by a failure initiated sending statelets,
8. of the two nodes terminating a restoration path, only the node with the smaller network ID initiated a loop-back test,
9. for a given statelet family, forward flooding was followed by reverse linking,
10. for a given statelet family, reverse linking was followed by a loop-back test, and
11. a transport signal was only restored after successfully completing a loop-back test.

Combined, all of these tests verify that OPRA behaves as described previously, lending credibility to the results presented in the following chapters.

# Chapter 11. OPRA Test Results

Having covered the theory of OPRA, the remainder of this thesis presents an analysis of OPRA's performance using the testbed explained in the last chapter. This chapter documents OPRA's ability to restore all single span failures in a metropolitan and a long haul network.

## 11.1 Test Networks

In chapter 4, six capacity placement techniques were introduced which either:

1. optimized the placement of *spare capacity* in a *span restorable network* (case 1),
2. optimized the placement of *spare capacity* in a *path restorable network without stub release* (case 2),
3. optimized the placement of *spare capacity* in a *path restorable network* with *stub release* (case 3),
4. optimized the placement of *working and spare capacity* in a *span restorable network* (case 4),
5. optimized the placement of *working and spare capacity* in a *path restorable network without stub release* (case 5), or
6. optimized the placement of *working and spare capacity* in a *path restorable network* with *stub release* (case 6).

Of the six capacity placement techniques for the five networks presented in chapter 4, and detailed in [38], OPRA's ability to restore all single span failures in a selection of 18 of those 30 designs was tested. This chapter is limited to detailing the results from 4 of those 18 tests in order to verify the key characteristics of OPRA without presenting an excessive amount of data. The conclusions drawn from the analysis of these four networks are common to all of the results obtained. A summary of the results obtained for the other 14 test networks is given at the end of this chapter.

In this chapter, OPRA's ability to restore all single span failures in a representative metropolitan network (network 2), and a representative long haul network (network 4), is investigated. The topologies of networks 2 and 4 are shown in chapter 4 in Figures 4.5 and 4.7 respectively. The capacity placement of networks 2 and 4 was designed using the IP presented in chapter 4 to either:

1. optimize the spare capacity placement of a path restorable network without stub release given that the working capacity design split the routing of demand between a node pair as evenly as possible over the node pair's logically shortest disjoint routes (i.e. case 2 as defined in chapter 4), or
2. optimize the placement of working and spare capacity of a path restorable network with stub release (i.e. case 6 as defined in chapter 4).

The test results from these four cases were chosen because OPRA is a path restoration mechanism, and of the four different path restorable capacity placement techniques presented in chapter 4, cases 2 and 6 represent the relevant limits of maximum sparing for operation of a path restorable mesh transport network.

## 11.2 Test Result Presentation Format

Span and network restorability, as well as restoration speed, are classified as operational metrics, and the first two plots presented for each network design quantify OPRA's performance using these operational metrics.

The first plot displays the restoration trajectory of all individual span failures in a network. For an example see Figure 11.1. Each trace plots percentage restoration (0% - 100%) against time. In addition to displaying each span's restoration status as a function of time, the first plot identifies:

1. each span's final restorability, $Rs_i$,
2. the final restorability of a network as a whole, Rn,
3. the lowest span restorability level of any span in the network, $Rn_{wc}$,
4. the time required to complete the first path of a restoration plan, $t_{p1}$,
5. the time required to complete the last path of a restoration plan, $t_R$,
6. the average time required to complete all restoration paths possible in a network, $t_{p,avg}$, and
7. the time at which 95% of all working paths are restored over all span cuts, $t_{95}$.

Given that the number of traces in each restoration trajectory plot equals the number of spans in a network, it can be difficult to distinguish the trace of one span failure from another. Consequently the trace of the span which experiences the shortest outage before achieving 100% restorability is highlighted in red, and the trace of the span which experiences the longest outage before achieving 100% restorability is highlighted in green. The traces corresponding to $Rn_{wc}$ and $t_R$ are also highlighted. While displaying all span restorability trajectories in a single graph makes distinguishing one trace from another difficult, it does present the performance envelope of when and how many spans are fully restored.

The second plot associated with each network design quantifies the statistics of an individual outage experienced by a restored working path, $t_{p,i}$. See Figure 11.2 for an example. A histogram which records the number of $t_{p,i}$ values which fall in a given 50 msec. window follows each restoration trajectory plot.

The remainder of the plots associated with each network quantify OPRA's performance using intrinsic path, rather than operational, metrics. Given that the networks used to test OPRA were all 100% restorable against individual span cuts, the span restorabilities of the restoration trajectory plot also quantify OPRA's Path Number Efficiency (PNE). The final restorability of each restoration trajectory in this plot implicitly compares the pathset found by OPRA to the reference pathset found by the IP, and may be interpreted as OPRA's routing efficiency or PNE, instead of the span restorability, for that span failure.

The network restorability of each network presented in this chapter was always very close to 100%. Networks deployed in practice will likely have more spare capacity than the tightly designed networks used to test OPRA because of the modularity of transmission systems, and OPRA would likely achieve 100% network restorability in such networks. Nonetheless, the Path Length Efficiency (PLE) of OPRA cannot be fairly determined here because the PNE of the reference IP solution (100%) and OPRA (97.9% - 99.7%) are not identical. Consequently, the distribution of the restoration path lengths found by OPRA and the IP are shown in this chapter in the third and fourth plots associated with each network instead. These two histograms facilitate comparing the restoration pathsets found by the IP and OPRA in the absence of a PLE value.

The fifth plot is a histogram of interference numbers. It captures for diagnostic interests only the degree of competition amongst statelets attempting to restore a span cut. For example, a small quantity of large interference numbers indicates most restoration paths found by OPRA don't prevent the formation of other restoration paths.

The final two plots associated with each network design compare a restoration path's interference number to the time the path was completed, and the path's length.

In summary, the following seven plots are presented for each of the four network designs discussed previously:

1. the restoration trajectories of all span cuts in terms of restoration status vs. time,
2. a histogram of restoration path completion times over all span cuts,
3. a histogram of restoration path lengths over all span cuts,

4. a histogram of restoration path lengths used in the reference IP solution over all span cuts,
5. a histogram of interference numbers over all span cuts,
6. a scatter plot relating interference numbers to restoration path completion times over all span cuts, and
7. a scatter plot relating interference numbers to restoration path lengths over all span cuts.

These plots are analysed in the discussion concluding this chapter.

## 11.3 Option Settings

For each of the test results presented in section 11.4 and 11.5, the values assigned to the various options defined in OPRA are listed below:

1. IIN_STEP = 0. When a unit of lost capacity is restored from a relation, the source from that relation increases the initial interference number (IIN) assigned to any statelets it initiates in subsequent broadcasts by IIN_STEP. Increasing the IIN of those statelets initiated by a source increases the likelihood that other demand pairs will complete one of their restoration paths. In a network which is not 100% restorable, the variable IIN_STEP helps achieve an overall pattern of individual restoration levels on each relation that is proportional to the network's pre-failure connectivity. Given that the networks used to test OPRA were 100% restorable against all single span cuts, IIN_STEP was set to zero when testing OPRA's ability to restore all single span cuts.

2. MAXREPEATS = 6 for network no. 2, and 11 for network no. 4. The maximum logical hop length of a restoration path found by OPRA is twice the value of MAXREPEATS. Each node traversed by a statelet increments that statelet's repeat field and rebroadcasts it except if MAXREPEATS would be exceeded. Any statelet that arrives at a node with a repeat value greater than or equal to MAXREPEATS is ignored. MAXREPEATS is set sufficiently large in all test cases to ensure that restoration paths at least as long as those specified in the reference IP solution are possible.

3. TIMEOUT = 3 seconds. The constant TIMEOUT signals the termination of a restoration event and is set sufficiently large to ensure OPRA is given enough time to find all restoration paths topologically feasible.

In addition to setting those options required to execute OPRA, the following values were assigned to the options associated with the testbed:

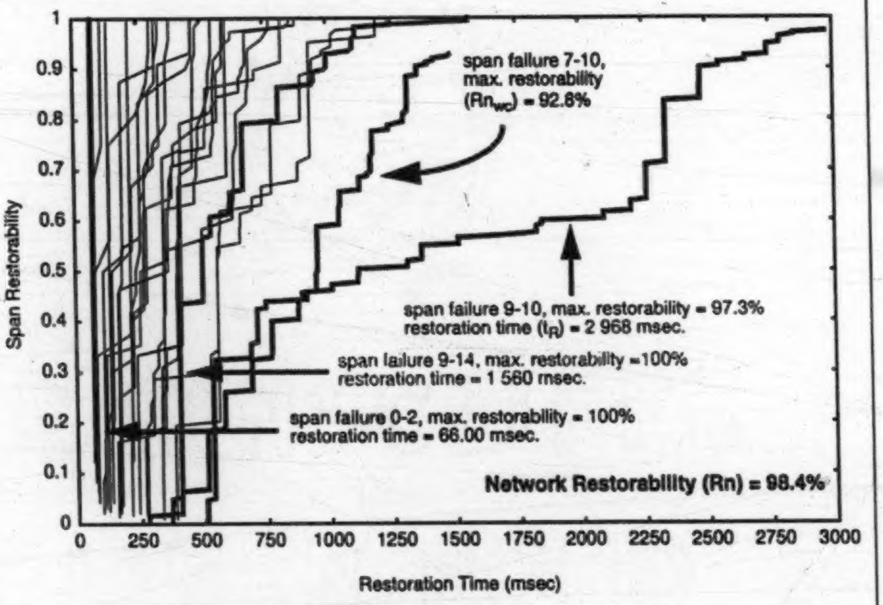
1. transmission delay = 1.25 msec.
2. propagation delay calculated assuming FOTS
3. processing delay associated with updating a statelet, STATELET_PROCESSING_DELAY, = 1 msec.
4. processing delay associated with computing the broadcast pattern at a node, BROADCAST_DELAY, = 2 msec.
5. time between polling cycles, DELTA_T, = 2 msec.

DELTA_T plus the processing delays is greater than the propagation and transmission delay combined, so restoration time is mainly determined by node delays in all of the test results presented here.

## 11.4 Experimental Results in a Metropolitan Network

### 11.4.1 Optimized Spare Capacity Design for a Path Restorable Network without Stub Release

The following plots record OPRA's performance in a metropolitan network (network number 2), with spare capacity optimized to facilitate path restoration without stub release (design case number 2).



span failure 7-10, max. restorability ($Rn_{wc}$) = 92.8%

span failure 9-10, max. restorability = 97.3% restoration time ($t_R$) = 2 968 msec.

span failure 9-14, max. restorability =100% restoration time = 1 560 msec.

span failure 0-2, max. restorability = 100% restoration time = 66.00 msec.

**Network Restorability (Rn) = 98.4%**

Note:

Time first alarm received at a node = 0 msec.
Time last alarm received at a node = 0 msec.

Time first restoration path found ($t_{p1}$) = 66.00 msec.
Time last restoration path found ($t_R$) = 2 968 msec.
Average restoration time ($t_{p,avg}$) = 655.6 msec.
Standard deviation from average restoration time = 552.8 msec.
Time required to find 0.95 of all restoration paths ($t_{95}$) = 2 335 msec.

Figure 11.1. Restoration trajectories in a metropolitan network without stub release

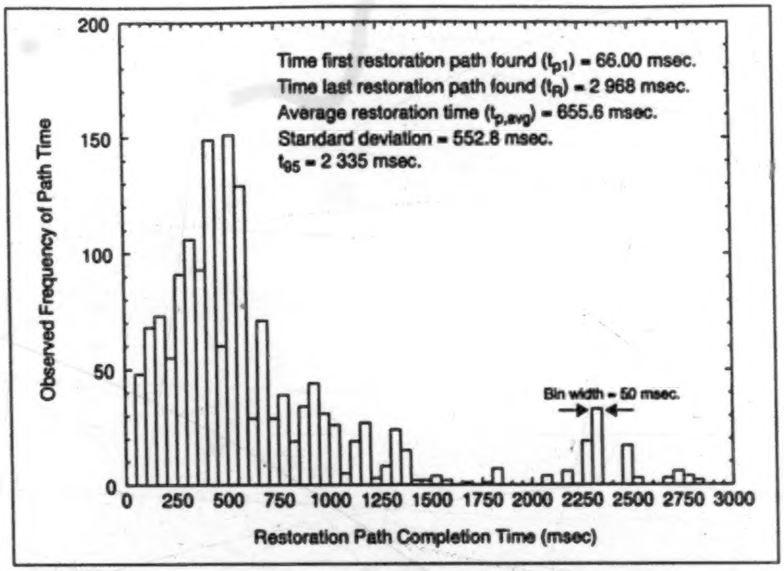


Figure 11.2. Distribution of restoration path times in a metropolitan network without stub release over all span cuts

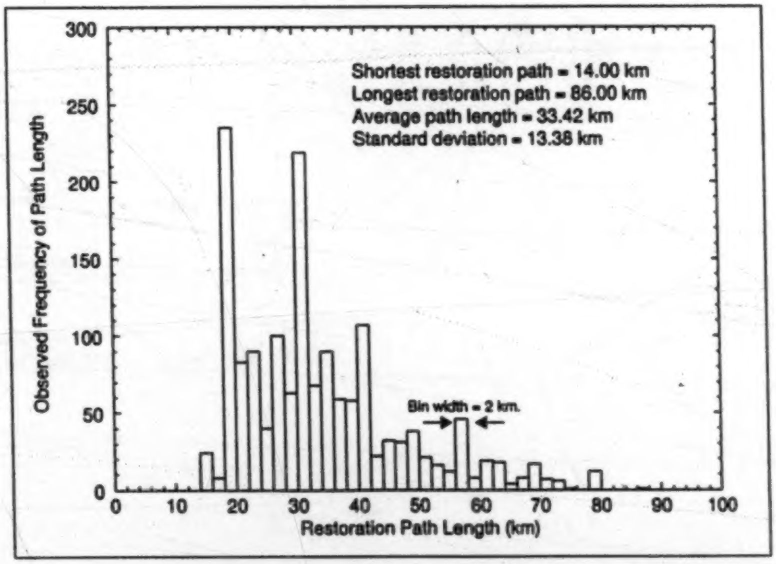


Figure 11.3. Distribution of restoration path lengths found by OPRA in a metropolitan network without stub release over all span cuts

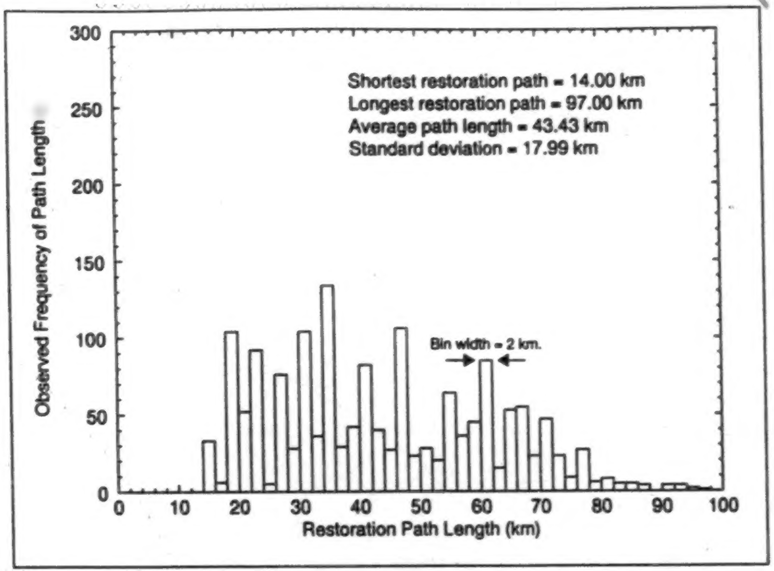


Figure 11.4. Distribution of restoration path lengths in the IP design for a metropolitan network without stub release over all span cuts

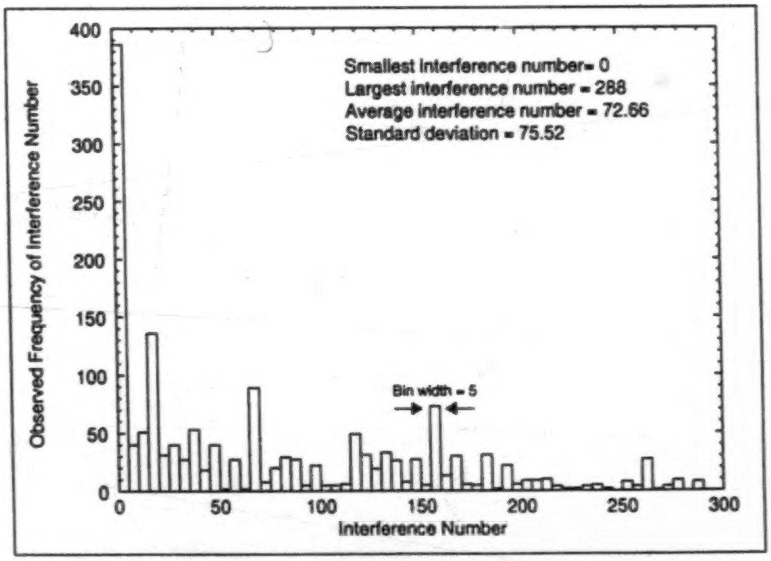


Figure 11.5. Distribution of interference numbers in a metropolitan network without stub release over all span cuts

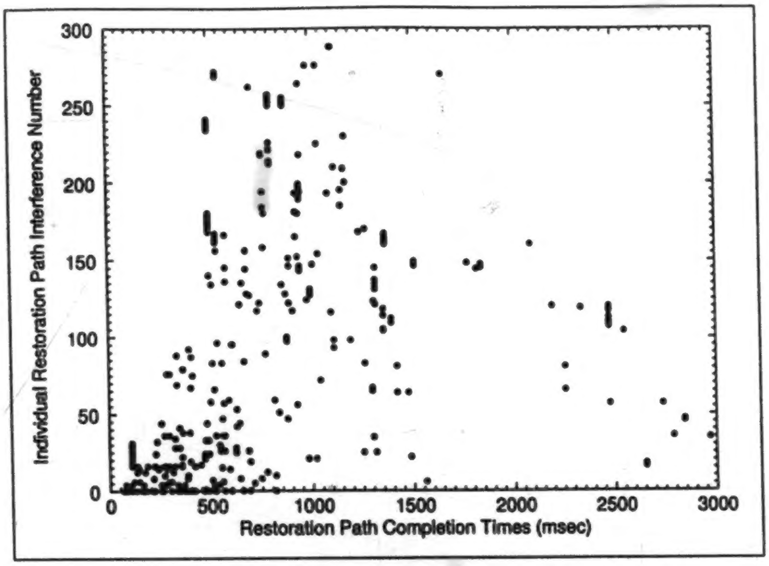


Figure 11.6. Interference number versus path time for all span cuts in a metropolitan network without stub release

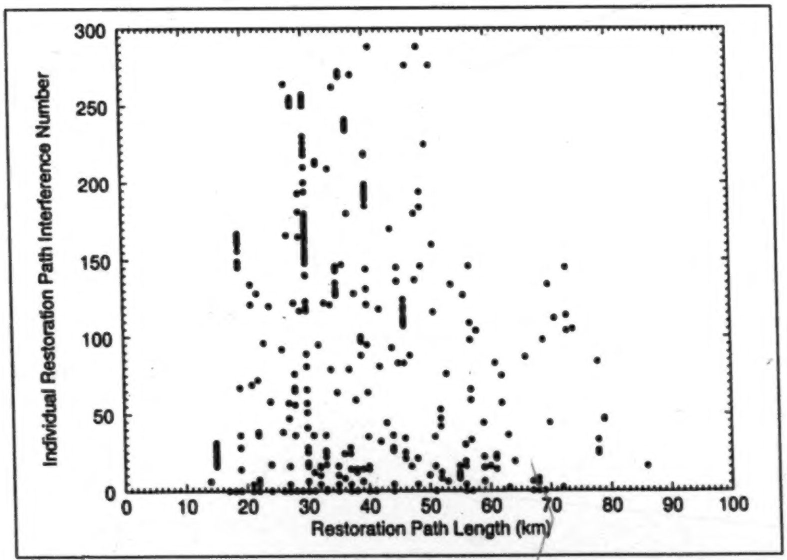
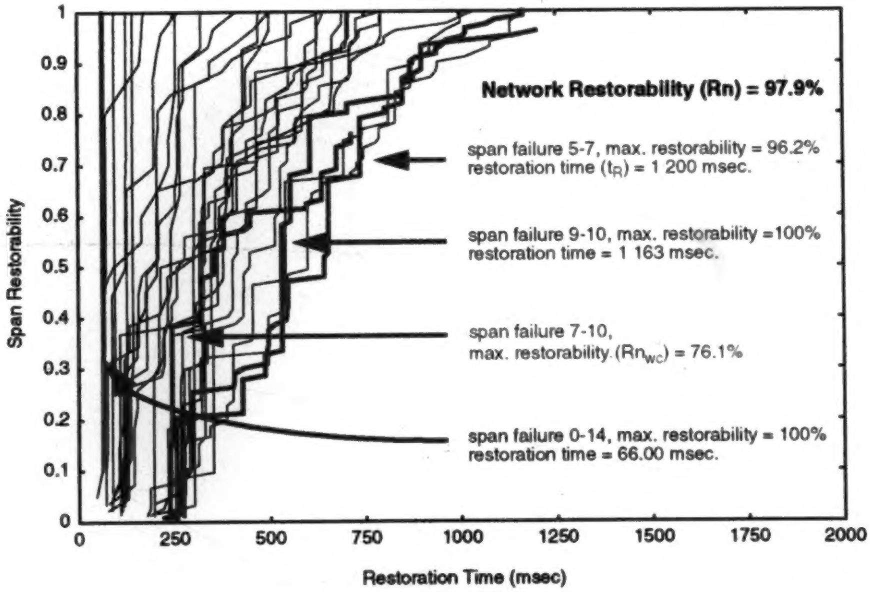


Figure 11.7. Interference number versus path length for all span cuts in a metropolitan network without stub release

### 11.4.2 Combined Capacity Design for a Path Restorable Network with Stub Release

The following plots display OPRA's performance in a metropolitan network (network number 2), with working and spare capacity optimized to facilitate path restoration with stub release (capacity design number 6). This may be considered the worst-case capacity minimum that a path restoration DRA could have to work with.



**Network Restorability (Rn) = 97.9%**

span failure 5-7, max. restorability = 96.2%
restoration time ($t_R$) = 1 200 msec.

span failure 9-10, max. restorability =100%
restoration time = 1 163 msec.

span failure 7-10,
max. restorability ($Rn_{wc}$) = 76.1%

span failure 0-14, max. restorability = 100%
restoration time = 66.00 msec.

**Note:**

**Time first alarm received at a node = 0 msec.**
**Time last alarm received at a node = 0 msec.**

**Time first restoration path found ($t_{p1}$) = 47.00 msec.**
**Time last restoration path found ($t_R$) = 1 200 msec.**
**Average restoration time ($t_{p,avg}$) = 402.2 msec.**
**Standard deviation from average restoration time = 215.8 msec.**
**Time required to find 0.95 of all restoration paths ($t_{95}$) = 851.0 msec.**

Figure 11.8. Restoration trajectories in a metropolitan network with stub release

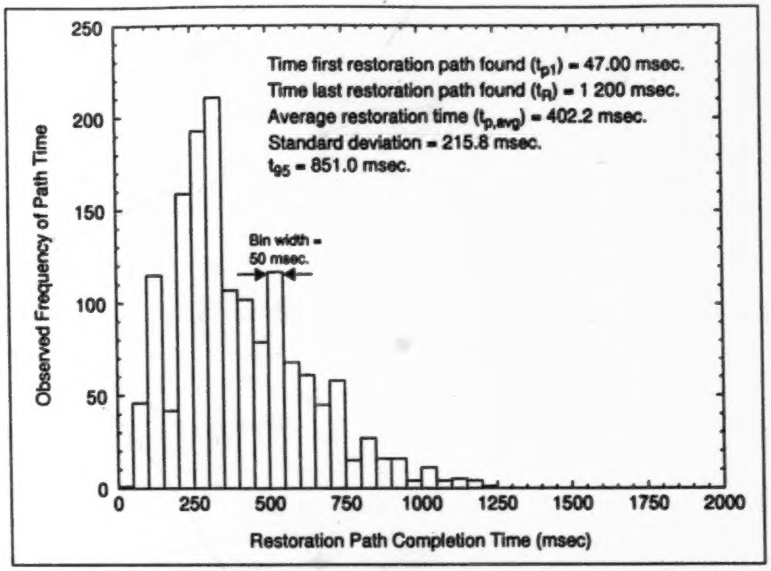


Figure 11.9. Distribution of restoration path times in a metropolitan network with stub release over all span cuts

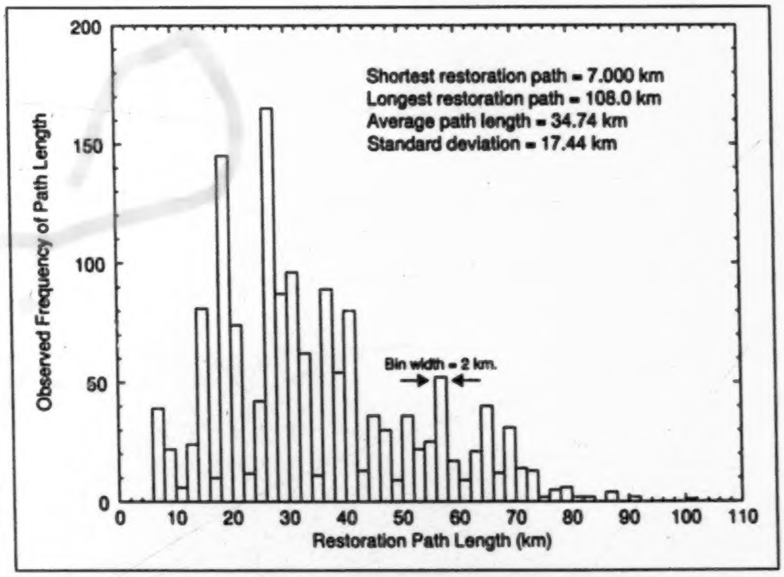


Figure 11.10. Distribution of restoration path lengths found by OPRA in a metropolitan network with stub release over all span cuts

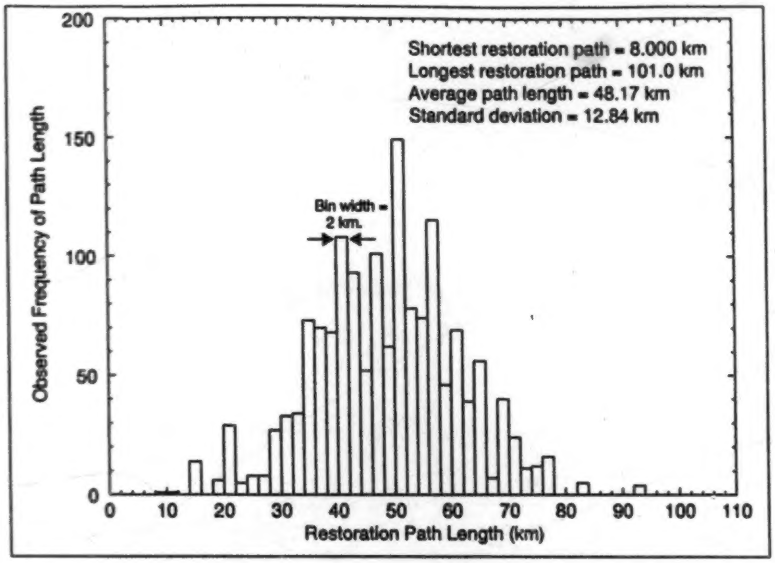


Figure 11.11. Distribution of restoration path lengths in the IP design for a metropolitan network with stub release over all span cuts

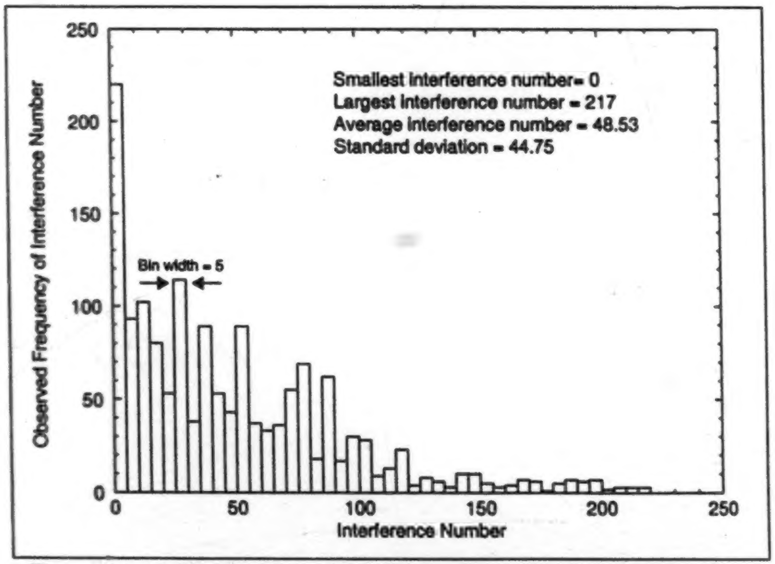


Figure 11.12. Distribution of interference numbers in a metropolitan network with stub release over all span cuts

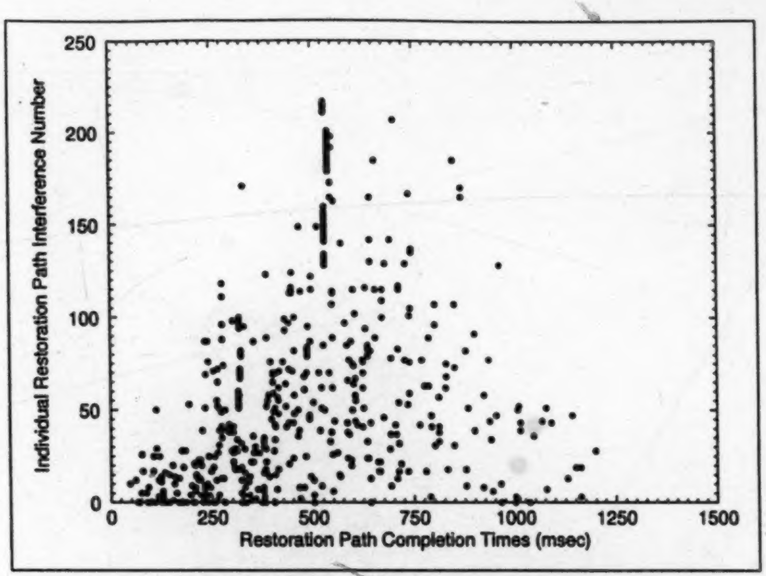


Figure 11.13. Interference number versus path time for all span cuts in a metropolitan network with stub release

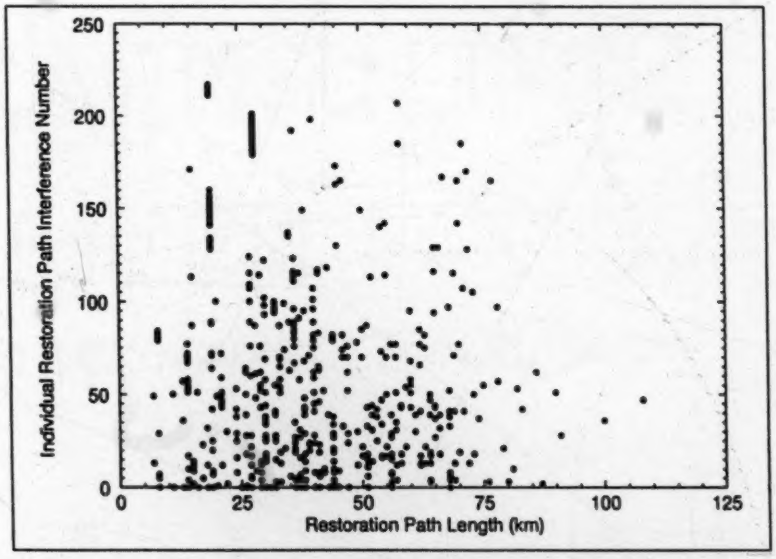
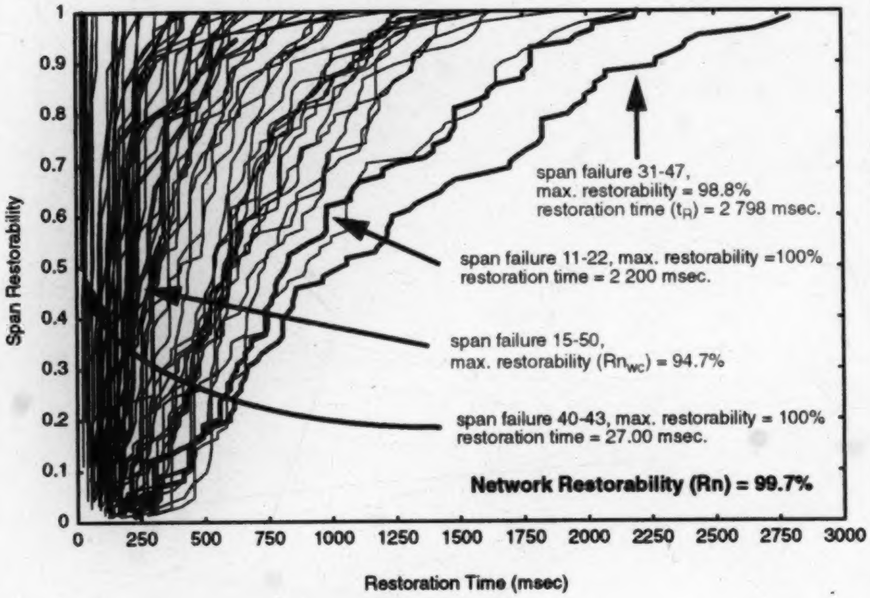


Figure 11.14. Interference number versus path length for all span cuts in a metropolitan network with stub release

## 11.5 Experimental Results in a Long Haul Network

### 11.5.1 Optimized Spare Capacity Design for a Path Restorable Network without Stub Release

The following plots show OPRA's performance in a long haul network (network number 4), with spare capacity optimized to facilitate path restoration without stub release (capacity design number 2).



**Note:**

**Time first alarm received at a node = 0 msec.**
**Time last alarm received at a node = 0 msec.**

**Time first restoration path found ($t_{p1}$) = 26.00 msec.**
**Time last restoration path found ($t_R$) = 2 798 msec.**
**Average restoration time ($t_{p,avg}$) = 521.2 msec.**
**Standard deviation from average restoration time = 406.8 msec.**
**Time required to find 0.95 of all restoration paths ($t_{95}$) = 1 391 msec.**

Figure 11.15. Restoration trajectories in a long haul network without stub release

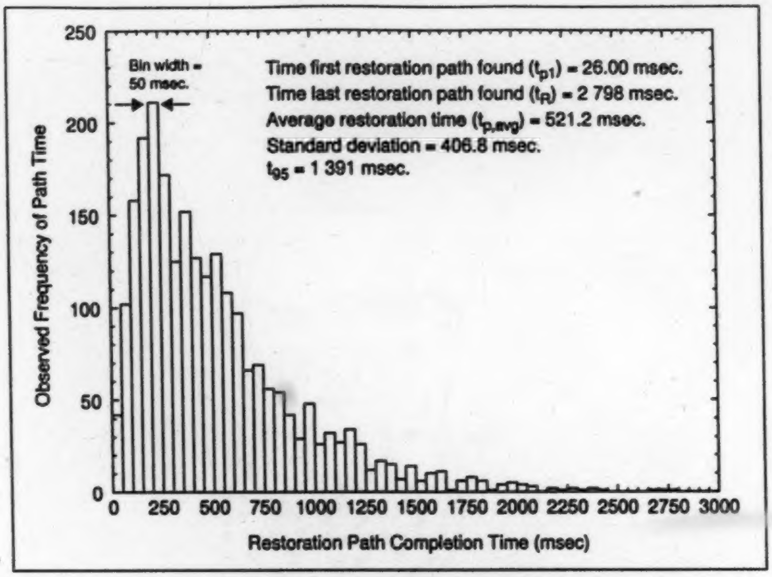


Figure 11.16. Distribution of restoration path times in a long haul network without stub release over all span cuts

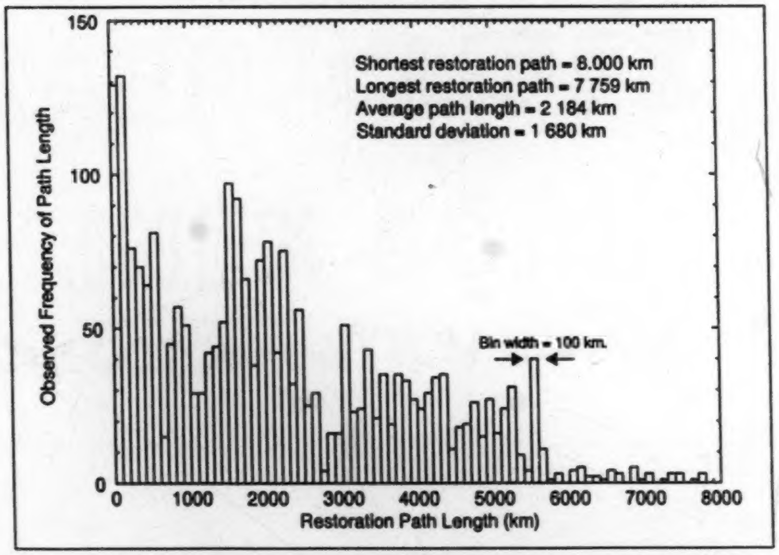


Figure 11.17. Distribution of restoration path lengths found by OPRA in a long haul network without stub release over all span cuts

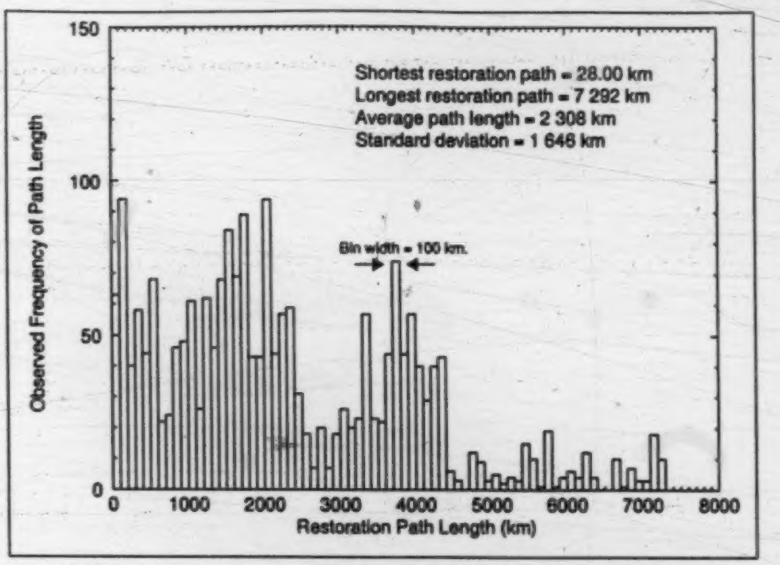


Figure 11.18. Distribution of restoration path lengths in the IP design for a long haul network without stub release over all span cuts

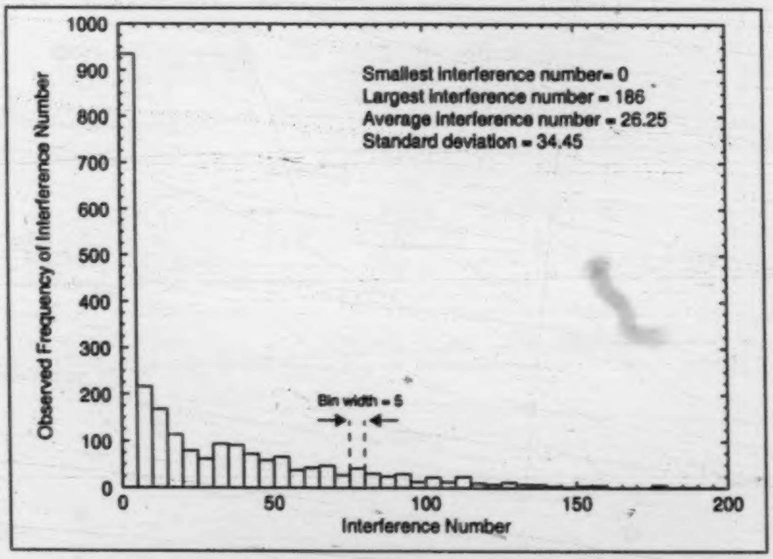


Figure 11.19. Distribution of interference numbers in a long haul network without stub release over all span cuts

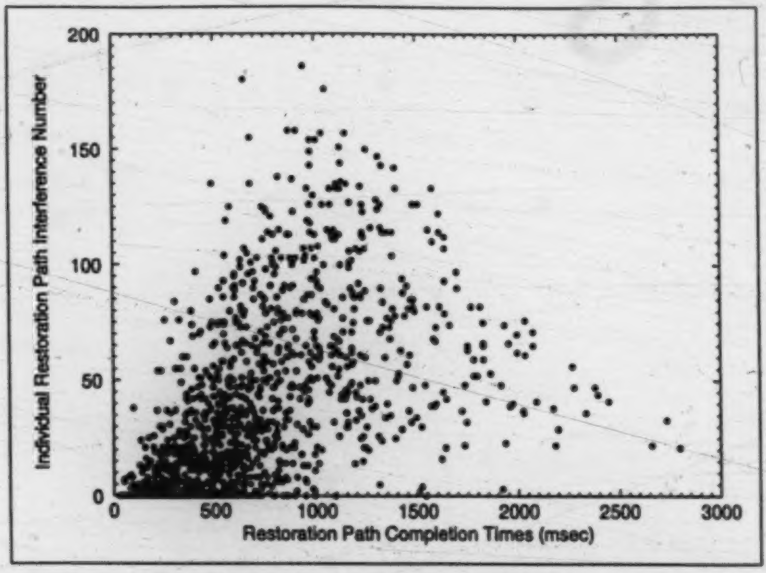


Figure 11.20. Interference number versus path time for all span cuts in a long haul network without stub release

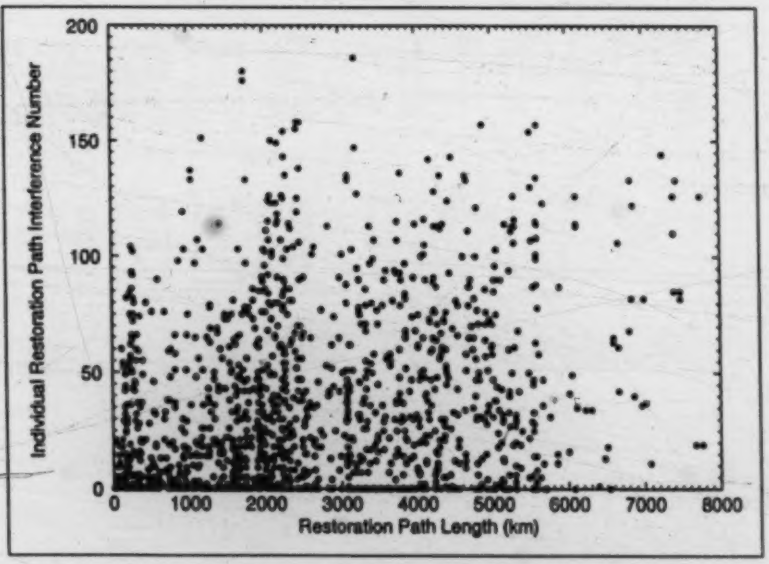
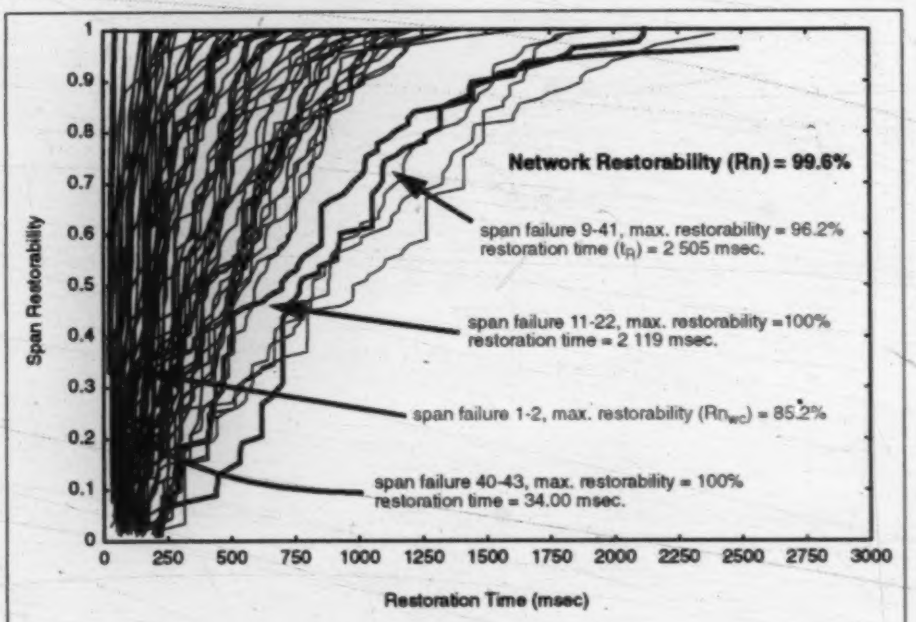


Figure 11.21. Interference number versus path length for all span cuts in a long haul network without stub release

### 11.5.2 Combined Capacity Design for a Path Restorable Network with Stub Release

The following plots record OPRA's performance in a long haul network (network number 4), with working and spare capacity optimized to facilitate path restoration with stub release (capacity design number 6).



**Note:**

Time first alarm received at a node = 0 msec.
Time last alarm received at a node = 0 msec.

Time first restoration path found ($t_{p1}$) = 18.00 msec.
Time last restoration path found ($t_R$) = 2 505 msec.
Average restoration time ($t_{p,avg}$) = 460.9 msec.
Standard deviation from average restoration time = 375.1 msec.
Time required to find 0.95 of all restoration paths ($t_{95}$) = 1 263 msec.

Figure 11.22. Restoration trajectories in a long haul network with stub release

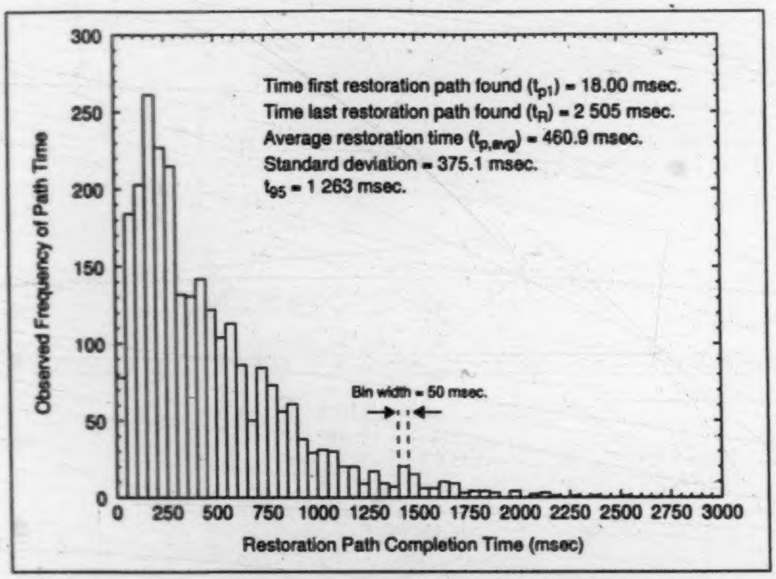


Figure 11.23. Distribution of restoration path times in a long haul network with stub release over all span cuts

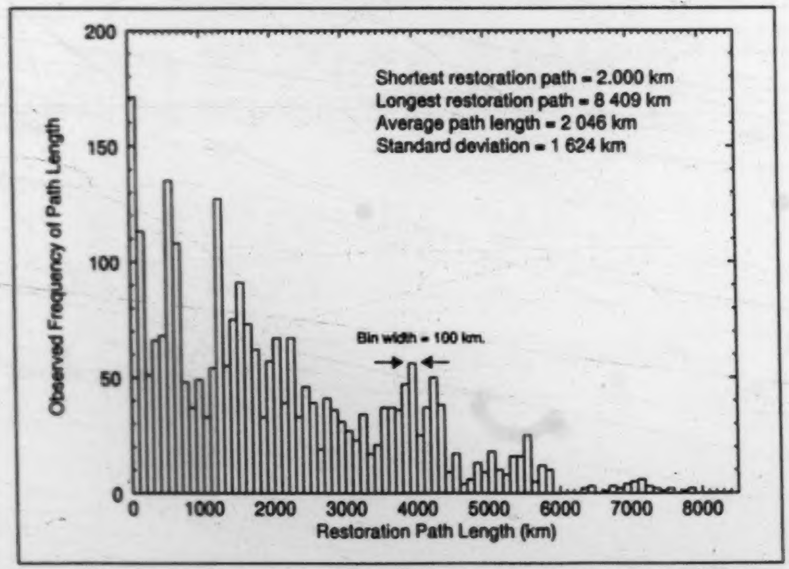


Figure 11.24. Distribution of restoration path lengths found by OPRA in a long haul network with stub release over all span cuts

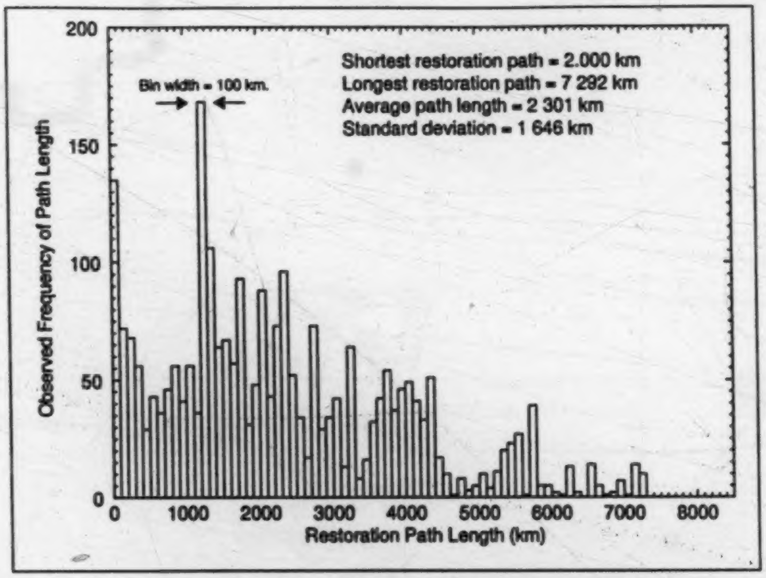


Figure 11.25. Distribution of restoration path lengths in the IP design for a long haul network with stub release over all span cuts

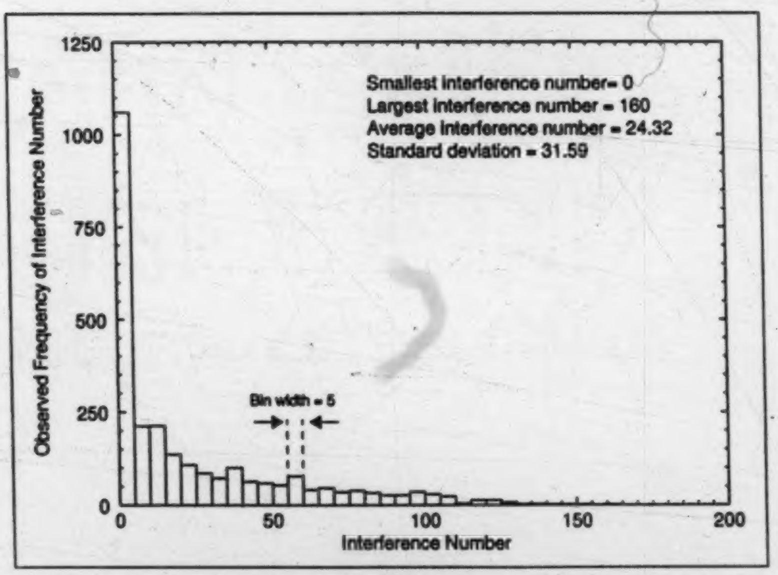


Figure 11.26. Distribution of interference numbers in a long haul network with stub release over all span cuts

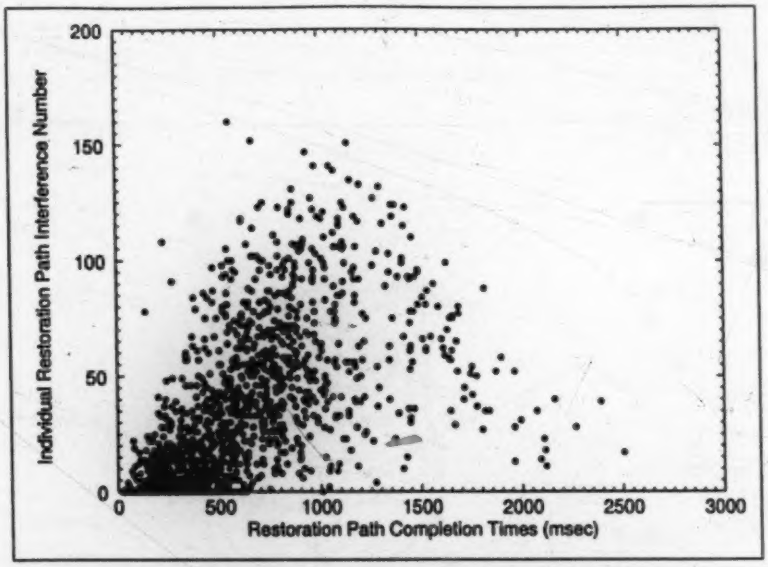


Figure 11.27. Interference number versus path time for all span cuts in a long haul network with stub release

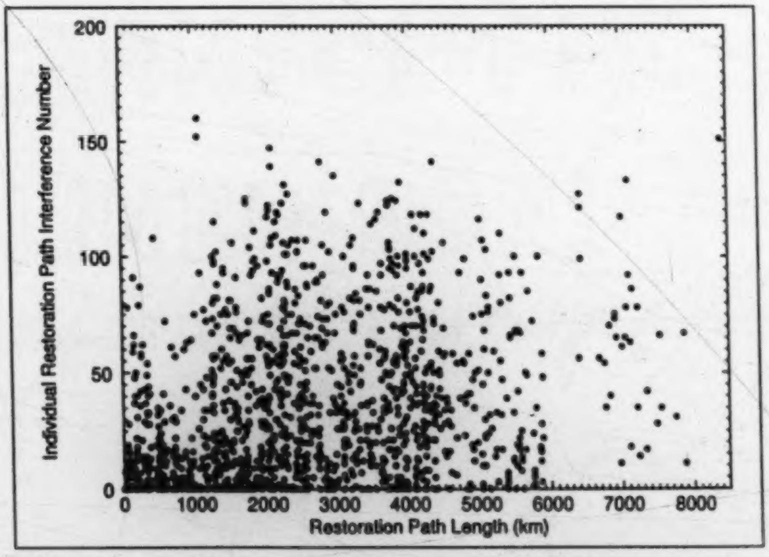


Figure 11.28. Interference number versus path length for all span cuts in a long haul network with stub release

## 11.6 Interpretation and Discussion of Results

The following tables summarize the results presented in the plots shown in the previous two sections and discussed in this section, as well as the characteristics of each test network.

Table 11.1. Operational performance metrics

| Network | Rn | $Rn_{wc}$ | $t_{p1}$ (msec) | $t_R$ (msec) | $t_{p,avg}$ (msec) | $t_{95}$ (msec) |
|---|---|---|---|---|---|---|
| 11.4.1 (metro network without stub release) | 98.4% | 92.8% | 66.00 | 2 968 | 655.6 | 2 335 |
| 11.4.2 (metro network with stub release) | 97.9% | 76.1% | 47.00 | 1 200 | 402.2 | 851.0 |
| 11.5.1 (long-haul network without stub release) | 99.7% | 94.7% | 26.00 | 2 798 | 521.2 | 1 391 |
| 11.5.2 (long-haul network with stub release) | 99.6% | 85.2% | 18.00 | 2 505 | 460.9 | 1 263 |

Table 11.2. Intrinsic path metrics

| Network | Avg. path length (OPRA) (km) | Standard deviation from avg. path length (OPRA) | Avg. path length (IP) (km) | Standard Deviation from avg. path length (IP) | Avg. Interference no. | Standard deviation from average interference no. |
|---|---|---|---|---|---|---|
| 11.4.1 (metro network without stub release) | 33.42 | 13.38 | 43.43 | 17.99 | 72.66 | 75.52 |
| 11.4.2 (metro network with stub release) | 34.74 | 17.44 | 48.17 | 12.84 | 48.53 | 44.75 |
| 11.5.1 (long-haul network without stub release) | 2 184 | 1 680 | 2 308 | 1 646 | 26.25 | 34.45 |
| 11.5.2 (long-haul network with stub release) | 2 046 | 1 624 | 2 301 | 1 646 | 24.32 | 31.59 |

Table 11.3. Test Network Characteristics

| Network | No. of nodes | No. of spans | Avg. network degree | Avg. span length (km) | Avg. no. of working links/span | Avg. no. of spare links/ span | Physical Redundancy |
|---|---|---|---|---|---|---|---|
| 11.4.1 (metro network without stub release) | 15 | 28 | 3.73 | 10.3 | 56.7 | 38.1 | 0.79 |
| 11.4.2 (metro network with stub release) | 15 | 28 | 3.73 | 10.3 | 55.0 | 21.2 | 0.37 |
| 11.5.1 (long-haul network without stub release) | 53 | 79 | 2.98 | 195 | 30.2 | 24.1 | 0.93 |
| 11.5.2 (long-haul network with stub release) | 53 | 79 | 2.98 | 195 | 33.4 | 16.2 | 0.50 |

### 11.6.1 Restoration Trajectories

From the four restoration trajectory plots it is apparent that OPRA is able to fully restore most span failures before the two second call dropping threshold. Network restorability, which here is the same as that network's PNE, ranges from 97.9% to 99.6%, with an average restoration time over all cases of 510 msec. In each case, those spans which OPRA fails to restore completely contain either the largest, or close to the largest, number of working links. In the metropolitan network, for both capacity designs presented, the last three spans to reach their final restorability are the three largest spans in the network. In Figure 11.1 these spans are 9-10, 7-10, and 9-14, with a final restorability of 97.3%, 92.8%, and 100% respectively, and working link counts of 262, 250, and 168 respectively.

At the other extreme, those spans which OPRA restores first contain the fewest, or close to the fewest, number of working links. The restoration trajectories for these small spans are often nearly vertical lines because all of the restoration paths required to restore these spans are identified at nearly the same instant as a result of the parallelism inherent in OPRA. In general, restoration trajectories appear as a staggered sequence of steps rather than a smooth trace because many statelets seek restoration paths concurrently and in parallel during the restoration process, resulting in bundles of restoration paths being found at discrete intervals in time as broadcast meshes from various statelet families grow and collapse in different regions of the network.

In each restoration trajectory plot, the average restoration time is close to 510 msec in spite of the large difference in size between the metropolitan and long haul examples. Given that the metropolitan network consists of 15 nodes and 22 spans, and the long haul network consists of 53 nodes and 79 spans, OPRA's ability to synthesize a near optimal restoration pathset quickly regardless of the number of spans and nodes in a network is apparent. This insensitivity to network size, where size is defined as the number of nodes and spans in a network, stems from the fact that a node executing OPRA only needs local knowledge of the links it terminates to compute that portion of the composite routing strategy which it is required to implement. Unlike centralized restoration, in distributed restoration the processing power of the platform on which an algorithm like OPRA runs grows as the size of the network grows, thereby maintaining a relatively constant restoration time regardless of the network's size, as long as node delays dominate.

With the conservative processing delays for the results in this chapter, a 100% complete restoration plan could only be identified before the two second call dropping threshold for the metropolitan network with stub release, as shown in Figure 11.8. However, each network's complete restoration plan was identified before three seconds, and the time required to find 95% of all restoration paths over all span cuts was less than two seconds in all but one case. In Figure 11.1, the 95th percentile is skewed by the trace of a single span failure. In Figure 11.1, the trace of span failure 9 -10 reaches a maximum restorability of 97.3%, 2.968 sec. after the arrival of the first path level alarm. Given that span 9 -10 is the largest span in this network with 262 links, the 95th percentile is raised to 2.335 sec even though every other span in the network is restored before two seconds. As shown in Figure 11.1, after a relatively quiet period centred around the 2 second mark, at which time the broadcast meshes of numerous statelet families competing to restore span failure 9 - 10 collapse, a final batch of restoration paths are found which raise span 9 - 10's restoration to 97.3%.

With the best processing delays possible in practice, it is likely that a complete restoration plan could be identified before the two second call dropping threshold in all of the tests presented here. As discussed in the following chapter, reducing the processing delay by a small amount, such that it remains realistic and comparable to the values chosen previously [21], the time required to complete the last path of the restoration plan of the network with the largest $t_R$ here, i.e. the metropolitan network without stub release, is reduced to less then two seconds.

Furthermore, as shown in Table 11.1, the Rn of each network was close, but not quite equal to 100%. It is expected that each network would be 100% restorable in practice because real networks will likely have somewhat more spare capacity than the tightly designed reference networks used here to test OPRA, due to the provisioning interval, and the modularity effects of transmission systems. This is apparent when an estimate of how much sparing needs to be added to each network in order to reach 100% restoration using OPRA is made. Such an estimate is possible by multiplying the total amount of unrestored demand in a network by the average restoration path length, and dividing that product by the total amount of spare capacity present in that network. For example, the metropolitan network with no stub release could not restore 25 units of demand. The average restoration path length in this network is 33.42 km as shown in Table 11.2. Therefore approximately 835.5 km of spare capacity are required to make this network 100% restorable using OPRA. This represents 3.7% of that network's total capacity. Similarly, the metropolitan network with stub release, the long-haul network with no stub release, and the long-haul network with stub release, would require approximately 5.5%, 0.97%, and 1.8% of that network's total capacity respectively to make them 100% restorable using OPRA.

### 11.6.2 Restoration Path Times

While all networks have similar average restoration times, $t_{p,avg}$, the time it takes on average to identify a restoration plan, $t_R$, in a network with fewer spares is generally less than in a network with many spares. This is evident when the histogram of restoration path times for those networks engineered using stub release is compared to those networks engineered without stub release. For both the metropolitan and long haul networks, when stub release is used, the IP requires fewer spare links to engineer a 100% path restorable network, and $t_R$, $t_{p,avg}$, and $t_{95}$ all decrease. In general, when a network has less spare capacity, the routing alternatives available to forward flooding statelets are reduced and OPRA is more constrained in its search for an optimal restoration pathset, thereby decreasing the restoration time. There is therefore a double benefit to having the most efficiently spared network design. The obvious benefit is reduced cost. Less obvious, but seen here, is increased speed of restoration.

### 11.6.3 Restoration Path Lengths

Comparing the restoration path length histograms from the solutions found using OPRA and the IP reference solutions, for each of the four network designs, it is apparent that OPRA tends to form shorter paths than the IP. For example, the average length of the restoration paths OPRA finds in the metropolitan network with stub release, as shown in Figure 11.10, is less than that found using the IP for that same network, as shown in Figure 11.11. In every case, OPRA restores a larger proportion of lost working paths using shorter restoration paths. However, synthesizing shorter restoration paths results in either synthesizing a few restoration paths longer than those used by the IP or a lower network restorability. The use of a few relatively long restoration paths to compensate for the use of a large number of relatively short restoration paths is apparent when the average and standard deviation shown in Figures 11.10 and 11.17 are compared to the average shown in Figures 11.11 and 11.18 respectively. In each of these cases, the average restoration path length found by OPRA is smaller than that of the IP, and the standard deviation larger.

Even though OPRA seeks to complete those restoration paths with the lowest interference number first, a forward flooding statelet with a larger interference number may succeed at establishing a short restoration path because low $t_{p,i}$'s as well as interference numbers moderate the competition between forward flooding statelets. Therefore, OPRA may establish a short restoration path with a larger interference number, instead of a more capacity efficient long restoration path with a lower interference number, because this is the minimal temporal path towards full restoration. The reference solution set found by the IP is only concerned with minimizing a network's capacity requirements, whereas OPRA's combined objective is to minimize both the restoration time and the spare capacity used. This leads OPRA to synthesize restoration paths generally shorter than those of the IP. In all the tests presented here, this trait reduces OPRA's PNE to slightly less than 100%, as shown in the network restorability figures associated with each restoration trajectory plot, which range from 97.9% to 99.7%.

### 11.6.4 Interference Numbers

The interference number histograms give an indication of the intensity of contention amongst statelets attempting to restore a span cut. For example, a histogram with a high average interference number indicates most restoration paths found by OPRA conflict with the formation of other possible restoration paths. In general, the number of statelets competing to restore a failure, and therefore each statelet's interference number, increases as a network's degree and spare capacity increases because such networks can support a greater number of statelet families simultaneously.

Comparing the interference number histograms for a given network, it is apparent that those histograms for networks employing stub release have a smaller average interference number and lower standard deviation than the histograms for networks without stub release. This is due to the fact that the amount of spare capacity available for restoration is less with stub release so that fewer statelet families compete to restore a failure simultaneously. Similarly, the average and standard deviation of the metropolitan network's interference number histograms are larger than those of the long haul network because the metropolitan network has a higher average node degree, and more statelet families compete to restore a failure simultaneously.

### 11.6.5 Scatterplots

OPRA seeks to complete those restoration paths with the lowest interference number first, and usually does so as shown in the path time versus interference number scatterplots, especially the scatterplots in Figures 11.20 and 11.27. The spread in the pattern shown in each of these plots is due to the fact that a forward flooding statelet with a larger interference number may succeed at establishing a restoration path because speed as well as interference numbers moderate the competition between forward flooding statelets.

Whether longer or shorter restoration paths are preferred by OPRA as a result of selecting those restoration paths with the lowest interference number first is revealed in the restoration path length versus interference number scatter plots. There is no apparent correlation between restoration path length and interference number. Long restoration paths are just as likely to have a large interference number as short restoration paths. Therefore, OPRA may synthesize a long or short restoration path with almost equal probability as a result of selecting those restoration paths with the lowest interference number first.

## 11.7 Summary of all OPRA Test Results

The following table summarizes OPRA's ability to restore all single span failures in the 18 network designs resulting from combining networks 1,2, and 4 with all six capacity placement techniques explained in chapter 4. The values presented in this summary table are consistent with the interpretation and discussion presented in the previous section.

One set of values shown in Table 11.4 is new however, and should be explained. The last column in Table 11.4 presents the number of unique statelets transmitted per link over all failures. This value is calculated by counting the total number of unique statelets transmitted on all the spans in a network during a single restoration event, and dividing that sum by the total number of spare links in that network. Only unique statelets are counted because OPRA only processes unique statelets as explained in chapter 10.

The average statelet volume per link may be decreased by bundling failed links from the same demand pair into a single group that is processed as one block at tandem nodes. In mesh restoration the most efficient use of spare capacity is made when every failed link unit is replaced by a replacement path of the same unit capacity. However, if larger units are re-routed as a single block average statelet volume per link is decreased [28]. Bundling is an implementation issue which is not investigated in this thesis.

Table 11.4. Summary of all OPRA Tests

| Network | IP capacity design case (as defined in chapter 4) | Rn | $t_{p1}$ (msec) | $t_R$ (msec) | $t_{p,avg}$ (msec) | Avg. interference no. | Avg. path length (km) | Average statelet volume per link* |
|---|---|---|---|---|---|---|---|---|
| 1 | Case 1 | 100% | 27.00 | 240.0 | 71.06 | 3.437 | 3.049 | 13.6 |
| | Case 2 | 100% | 25.00 | 251.0 | 77.89 | 4.246 | 3.169 | 15.7 |
| | Case 3 | 100% | 28.00 | 235.0 | 78.66 | 4.880 | 3.239 | 17.3 |
| | Case 4 | 100% | 27.00 | 206.0 | 69.20 | 5.218 | 2.859 | 16.6 |
| | Case 5 | 97.2% | 28.00 | 243.0 | 82.12 | 4.275 | 3.319 | 16.4 |
| | Case 6 | 98.6% | 27.00 | 266.0 | 81.48 | 3.936 | 3.464 | 17.9 |
| 2 | Case 1 | 99.6% | 70.00 | 1 999 | 625.9 | 52.54 | 32.36 | 6.23 |
| | Case 2 | 98.4% | 66.00 | 2 968 | 655.6 | 72.66 | 33.42 | 7.84 |
| | Case 3 | 98.0% | 62.00 | 1 959 | 543.3 | 65.27 | 33.12 | 7.38 |
| | Case 4 | 98.9% | 52.00 | 1 381 | 410.5 | 45.69 | 35.27 | 9.78 |
| | Case 5 | 99.7% | 54.00 | 1 135 | 410.0 | 51.39 | 36.86 | 11.4 |
| | Case 6 | 97.9% | 47.00 | 1 200 | 402.2 | 48.53 | 34.74 | 12.3 |
| 4 | Case 1 | 99.5% | 24.00 | 2 876 | 546.3 | 19.89 | 2 161 | 11.8 |
| | Case 2 | 99.7% | 26.00 | 2 798 | 521.2 | 26.25 | 2 184 | 24.5 |
| | Case 3 | 99.7% | 23.00 | 2 774 | 527.4 | 25.27 | 2 199 | 26.0 |
| | Case 4 | 99.1% | 15.00 | 2 553 | 481.5 | 21.00 | 2 203 | 20.5 |
| | Case 5 | 99.3% | 21.00 | 2 725 | 496.6 | 24.67 | 2 216 | 31.0 |
| | Case 6 | 99.6% | 18.00 | 2 505 | 460.9 | 24.32 | 2 046 | 31.5 |

* The average statelet volume counts the number of *unique* statelets transmitted per link over all failures.

## 11.8 Conclusion

OPRA restores all single span failures in a network relatively quickly and efficiently. Given the conservative processing delays and tightly designed networks used to find the results presented in this chapter, a complete restoration plan which fully restores any single span failure would likely be identified before the two second call dropping threshold in each of these networks in practice.

In all of the tests presented in this chapter, a network's restorability and the PNE of the process was never less than 97.9% and the average restoration time never more than 656 msec. The difference in PNE between the solutions found by the IP and OPRA may be attributed in part to OPRA's combined objective to minimize both the restoration time, and the spare capacity used restoring a failure.

At the heart of OPRA is the interference heuristic through which the DRA self-organizes network spares into a near optimal multicommodity max-flow restoration path-set. In accordance with this heuristic, those paths with the lowest interference number are generally found first. Unlike a path's restoration time and its interference number, there is no correlation between a restoration path's length and its interference number. Therefore OPRA may synthesize long as well as short restoration paths.

While the topology and size of any network designed to survive all single span failures have little impact on the restoration time and PNE, they impact a restoration path's interference number. More constrained networks with less spare capacity and lower average node degree cannot simultaneously support as many statelet families as networks which have more spare capacity and a higher average degree, and therefore the restoration paths of these more constrained networks have lower interference numbers.

The consistency of OPRA's restoration time and PNE in all of the tests performed demonstrates the algorithm's flexibility and insensitivity to network size and topology.

# Chapter 12. Effects of Decreasing the Processing Delay

This chapter investigates the impact of reducing the processing delay associated with updating ports, re-evaluating the broadcast pattern at a node, and the delay between polling cycles. The time required to process a new statelet or alarm is set by the constant STATELET_PROCESSING_DELAY. The time required to re-evaluate the broadcast pattern at a node is set by the constant BROADCAST_DELAY. The delay between polling cycles is set by the constant DELTA_T. In the previous chapter, the results presented assume a STATELET_PROCESSING_DELAY of 1 msec, a BROADCAST_DELAY of 2 msec, and a DELTA_T of 2 msec. Given these conservative processing delays a complete restoration plan could only be identified before the two second call dropping threshold for the metropolitan network with stub release.

In this chapter the impact of reducing STATELET_PROCESSING_DELAY to 0.5 msec, BROADCAST_DELAY to 1 msec, and DELTA_T to 0.5 msec is investigated. Given a DCS CPU clock rate of 200 MHz, assuming a C-level instruction takes 5 clock cycles to complete, and that processing one port requires in the worst case on the order of a few hundred lines of C-code, the processing delays considered in this chapter remain realistic and comparable to the values chosen previously [21].

## 12.1 Study Network

The effect of reducing the processing delay in the network with the largest 95th percentile restoration time is presented in this chapter. The metropolitan network without stub release required 2 335 msec to find 0.95 of all restoration paths over all span cuts, and is used here to investigate the impact of reducing the processing delay associated with updating ports, re-evaluating the broadcast pattern at a node, and the delay between polling cycles. This network optimizes the spare capacity placement of a path restorable network without stub release given that the working capacity design split the routing of demand between a node pair as evenly as possible over the node pair's logically shortest disjoint routes (i.e. case 2 as defined in chapter 4).

## 12.2 Test Result Presentation Format

The same set of plots presented in chapter 11 for each network design are presented here for the metropolitan network without stub release, except for the histogram of restoration path lengths used in the reference IP solution over all span cuts because it doesn't change.

## 12.3 Option Settings

The same values assigned in chapter 11 to the various options defined in OPRA are reused here. In summary they are:

IIN_STEP = 0
MAXREPEATS = 6
TIMEOUT = 3 sec.

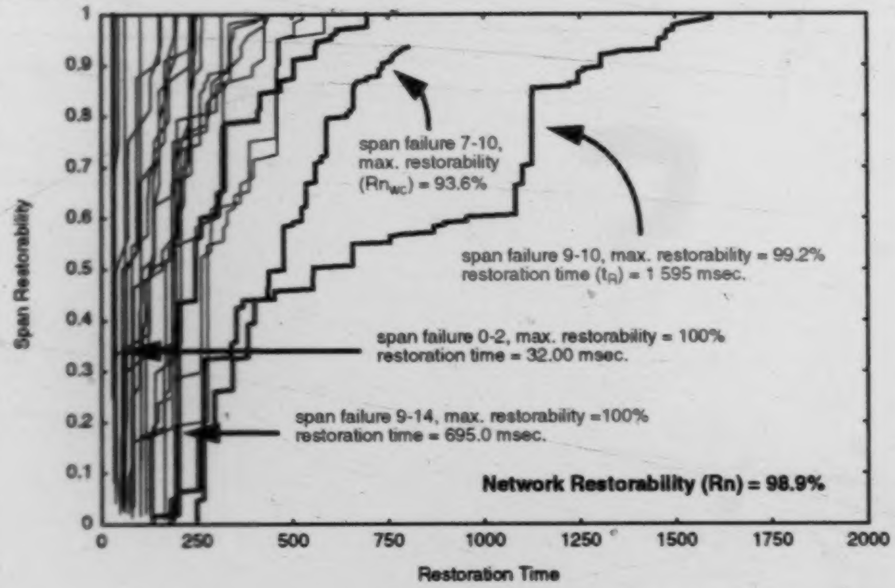
Retaining these same option values facilitates a fair comparison between the set of plots presented in chapter 11 for the metropolitan network without stub release and the plots presented in this chapter.

In order to investigate the impact of reducing the processing delay, the values assigned to the options associated with the testbed are different in this chapter than in chapter 11. In this chapter the following values were assigned to the options associated with the testbed:

1. transmission delay = 1.25 msec.
2. propagation delay calculated assuming FOTS
3. STATELET_PROCESSING_DELAY, = 0.5 msec.
4. BROADCAST_DELAY = 1 msec.
5. DELTA_T = 0.5 msec.

## 12.4 Experimental Results in Metropolitan Network without Stub Release

The following plots record OPRA's performance in a metropolitan network (network number 2), with spare capacity optimized to facilitate path restoration without stub release (design case number 2) with a reduced processing delay.



Note:

Time first alarm received at a node = 0 msec.
Time last alarm received at a node = 0 msec.

Time first restoration path found $(t_{p1})$ = 32.00 msec.
Time last restoration path found $(t_R)$ = 1 595 msec.
Average restoration time $(t_{p,avg})$ = 330.3 msec.
Standard deviation from average restoration time = 283.7 msec.
Time required to find 0.95 of all restoration paths $(t_{95})$ = 1 100 msec.

Figure 12.1. Restoration trajectories in a metropolitan network without stub release

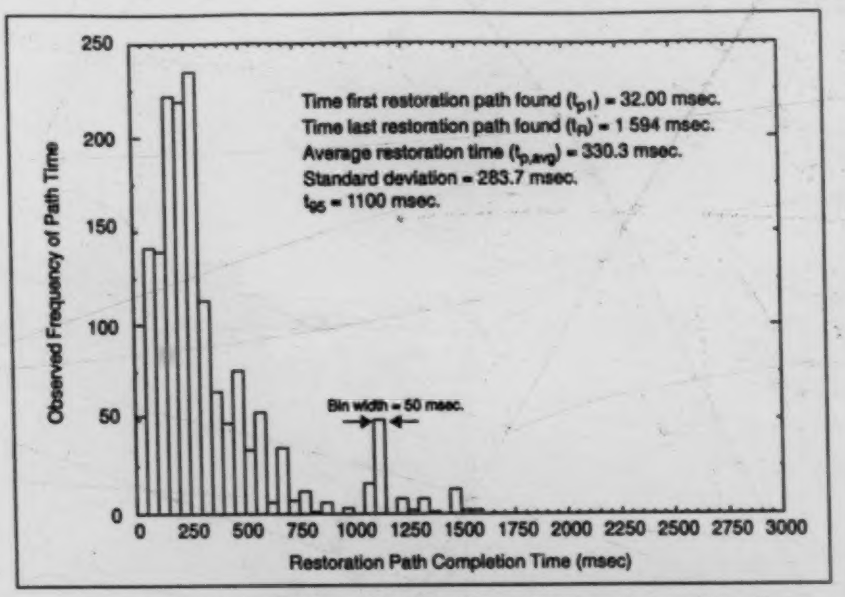


Figure 12.2. Distribution of restoration path times in a metropolitan network without stub release over all span cuts

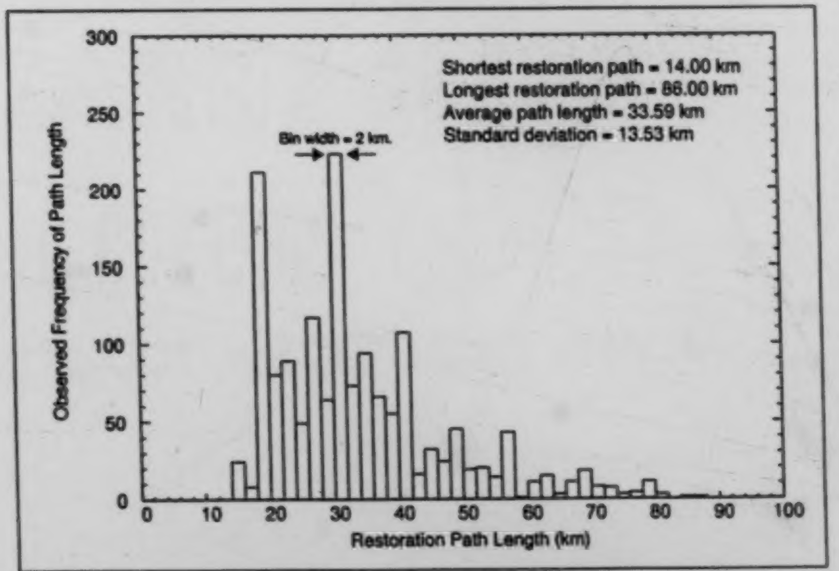


Figure 12.3. Distribution of restoration path lengths found by OPRA in a metropolitan network without stub release over all span cuts

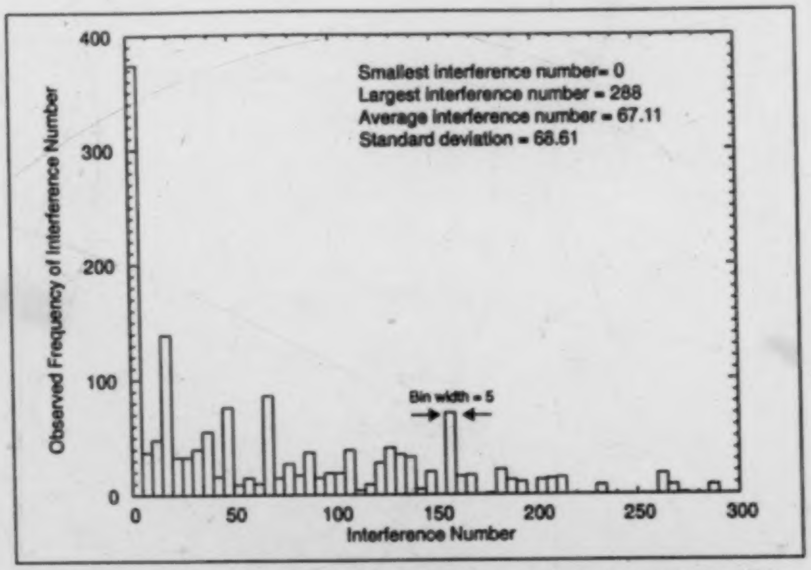


Figure 12.4. Distribution of interference numbers in a metropolitan network without stub release over all span cuts

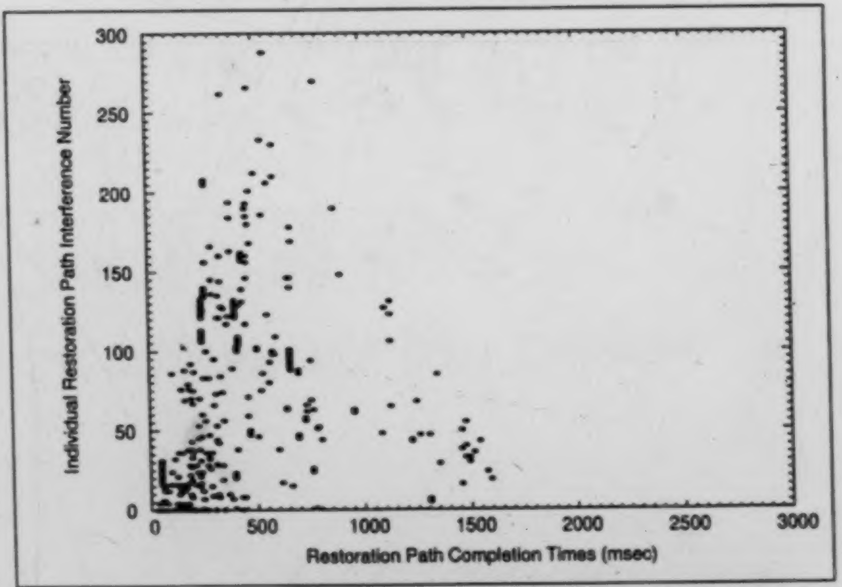


Figure 12.5. Interference number versus path time for all span cuts in a metropolitan network without stub release

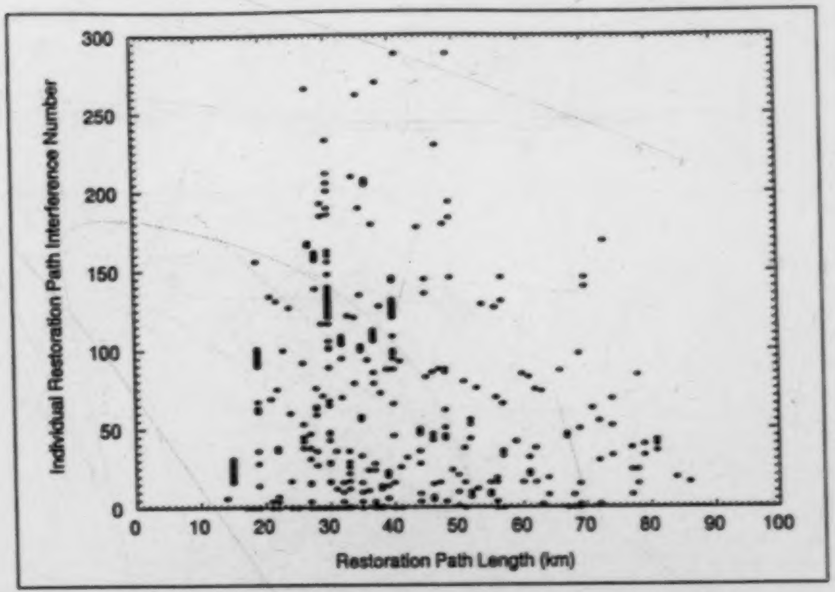


Figure 12.6. Interference number versus path length for all span cuts in a metropolitan network without stub release

## 12.5 Interpretation of Results

### 12.5.1 Restoration Trajectories

Comparing the restoration trajectories presented in Figure 12.1 to those of the same network in Chapter 11, it is evident that:

1. reducing the processing delay associated with updating ports from 1 msec. to 0.5 msec,
2. reducing the processing delay associated with re-evaluating the broadcast pattern from 2 msec to 1 msec, and
3. reducing the delay between polling cycles from 2 msec. to 0.5 msec,

nearly cuts the restoration time in half. The average restoration time moved from 655.6 msec. to 330.2 msec, and the time required to find 0.95 of all restoration paths decreased from 2 335 msec. to 1 100 msec.

Reducing the delay also increased network restorability slightly. Network restorability moved from 98.4% to 98.9%. Furthermore, the restorability of those spans with the lowest restorability, $Rn_{wc}$, and restored last, $t_R$, increased slightly from 92.8% and 97.3%, to 93.6% and 99.2% respectively.

Reducing the processing delay has little effect on which spans have the lowest and highest restoration times, and which spans have the lowest restorability. In both Figures 11.1 and 12.1, span failure 0-2 has the lowest restoration time, span failure 9-10 has the longest restoration time, and span failure 7-10 has the lowest restorability.

### 12.5.2 Restoration Path Times

Comparing histograms 11.2 and 12.2 it is apparent that the restoration time is almost cut in half in each of $t_{p1}$, $t_R$, $t_{p,avg}$, and $t_{95}$. However, the overall distribution of the histogram is the same in both plots.

### 12.5.3 Restoration Path Lengths

Comparing histograms 11.3 and 12.3 it is apparent that the distribution of path lengths found by OPRA does not change very much. The length of the shortest and longest paths are the same in both plots, and the average restoration path length and standard deviation are almost equal.

### 12.5.4 Interference Numbers

Comparing histograms 11.5 and 12.4 it is apparent that the distribution of interference numbers does not change very much either. The smallest and largest interference numbers in both plots have the same value, and the average interference number and standard deviation are almost equal.

### 12.5.5 Scatterplots

Comparing Figures 11.6 and 12.5 it is apparent that reducing the processing delay has compressed the plot of interference numbers versus path completion times along the horizontal time axis. This effect is expected because reducing the processing delay reduces each path's restoration time. The scatterplots are similar in all other respects. Those restoration paths with the lowest interference number are still usually restored first, and the spread in the pattern shown is due to the fact a forward flooding statelet with a larger interference number may succeed at establishing a restoration path.

Comparing Figures 11.7 and 12.6 it is evident that reducing the processing delay has little impact on the plot of path length versus interference number. In both scatterplots there is no discernible correlation between a restoration path's length and its interference number.

## 12.6 Conclusion

Reducing the processing delay reduces the restoration time and has little impact on a network's restorability. Given the realistic processing delays and the tightly designed network used to find the results presented in this chapter, it is likely that a complete restoration plan which fully restores any single span failure would be identified by OPRA before the two second call dropping threshold in any network in practice.

# Chapter 13. Effects of Random Individual Link Failure Times

This chapter reports the behaviour of OPRA when presented with a random series of individual link failures, as opposed to a "guillotine" (i.e. simultaneous) model of a cable cut like the one used in the previous two chapters. A staggered series of individual link alarms seems more plausible when a backhoe pulls a cable apart than the idealization of an instantaneous complete failure, and verifies OPRA's ability to cope with complex failure stimuli.

## 13.1 Staggered Alarms

Given that OPRA reconfigures the spares of a network in 100's of milliseconds, a network failure may be ongoing while the restoration process is active. Random link failures spread out over a few hundred milliseconds could, in principle, offer the greatest potential for possibly adverse dynamic interactions. This chapter therefore investigates the effects of individual alarms that are randomly distributed over an interval of several hundred milliseconds on the behaviour of OPRA, similar to the investigations performed previously on the SHN [32].

A Gaussian distribution of random failure times was used to produce staggered alarm failure scenarios as in [32]. For the results presented here, individual link failure times have a standard deviation of 96.87 msec about an observed mean of 209.9 msec. This mean and standard deviation was chosen because it coincides with the time scale of a single restoration event. Non-simultaneous link failures spaced at intervals equal to, or longer than, a few seconds would be restored by OPRA in separate distinct sequential restoration events. Such a test would be equivalent to running multiple independent sequential tests of OPRA in a network with steadily diminishing spare capacity, and not clearly verify OPRA's ability to cope with complex failure stimuli.

The assignment of failure times to individual links within the cross-section of a failing span is made randomly, and without regard to working or spare status, so that the overall span-cutting action appears random both in space and in time.

## 13.2 Direct Exposure of OPRA to Staggered Alarms

The results presented in this chapter assume OPRA is directly exposed to a series of randomly timed individual link failures. In this case OPRA is re-invoked in response to each new failure by a 'late-alarm' event while already active in response to previous failures. Each such re-invocation causes a source node to update its primary flooding pattern, and such changes have to propagate through the rest of the network, altering the self-organizing dynamics of the network on-the-fly.

An implication of direct exposure of OPRA to staggered alarms is that a spare link on a failing span may be used for restoration and then fail itself later. The ability to restore over spare links on the same span as the failure is normally desirable because it means that single fibre failures are restored in a manner analogous to today's 1:N or 1:1 automatic protection switches (APS).

## 13.3 Study Network

The network used for the study of staggered alarms is the metropolitan network with stub release investigated in chapter 11. This network has its working and spare capacity optimized for path restoration with stub release (i.e. case 6 as defined in chapter 4). Of all the networks investigated in chapter 11 assuming a guillotine cable cut, the metropolitan network with stub release was the only network for which OPRA identified a complete restoration plan before the two second call dropping threshold. This network is used here because any detrimental effects from staggered alarms on restoration time and restorability will be clearest in this network when compared to the benchmark of the simultaneous failure case presented in chapter 11.

## 13.4 Test Result Presentation Format

The same set of plots presented in chapters 11 and 12 are presented here for the metropolitan network with stub release. In addition, a histogram of working link failure times is presented.

## 13.5 Option Settings

The same values assigned in chapter 11 to the various options defined in OPRA and the testbed are reused here. In summary they are:

IIN_STEP = 0
MAXREPEATS = 6
TIMEOUT = 3 sec.
transmission delay = 1.25 msec.
propagation delay calculated assuming FOTS
STATELET_PROCESSING_DELAY, = 1 msec.
BROADCAST_DELAY = 2 msec.
DELTA_T = 2 msec.

## 13.6 Experimental Results in Metropolitan Network with Stub Release

The following plots show OPRA's performance in a metropolitan network (network number 2), with working and spare capacity optimized to facilitate path restoration with stub release (design case number 6) under staggered alarm conditions.

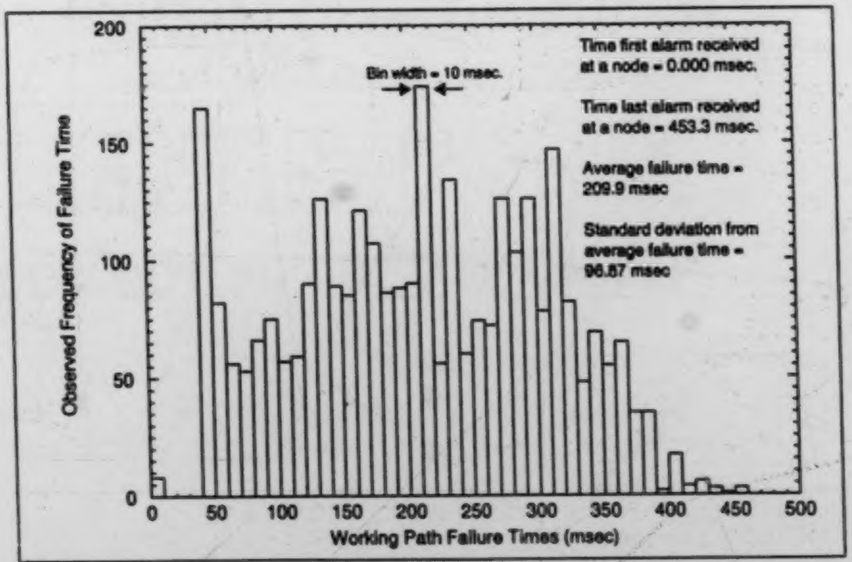
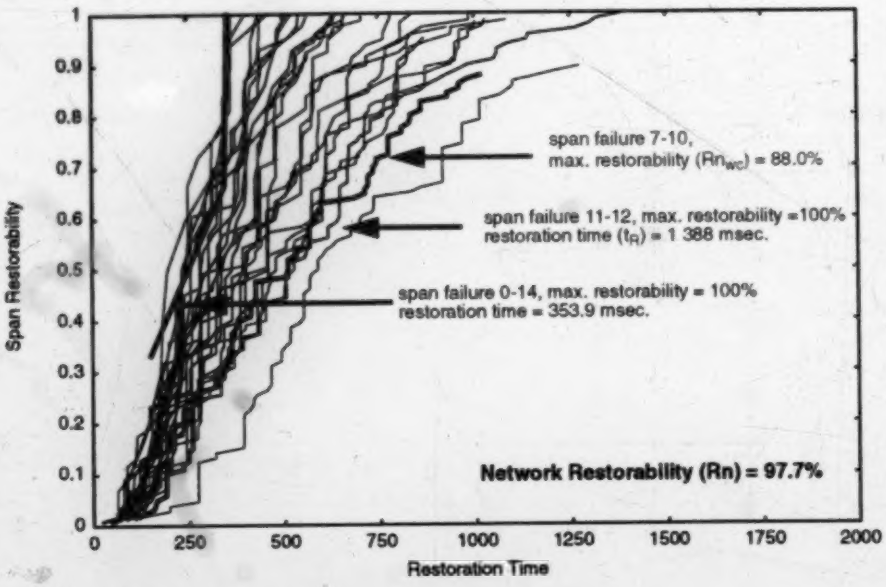


Figure 13.1. Distribution of alarms in a metropolitan network over all span cuts



Note:

Time first alarm received at a node = 0.000 msec.
Time last alarm received at a node = 453.3 msec.
Average failure time = 209.9 msec
Standard deviation from average failure time = 96.87 msec

Time first restoration path found ($t_{p1}$) = 26.00 msec.
Time last restoration path found ($t_R$) = 1 388 msec.
Average restoration time ($t_{p,avg}$) = 423.5 msec.
Standard deviation from average restoration time = 233.4 msec.
Time required to find 0.95 of all restoration paths ($t_{95}$) = 916.0 msec.

Figure 13.2. Restoration trajectories in a metropolitan network with stub release

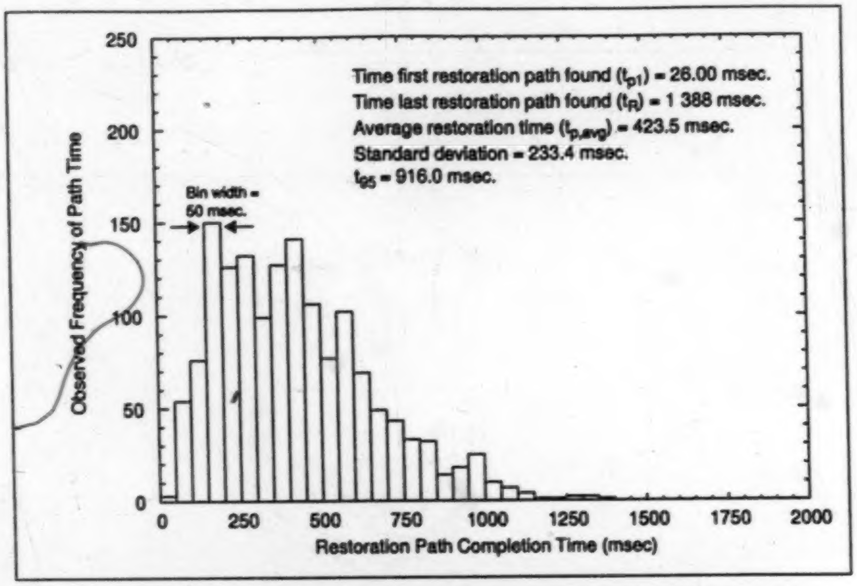


Figure 13.3. Distribution of restoration path times in a metropolitan network with stub release over all span cuts

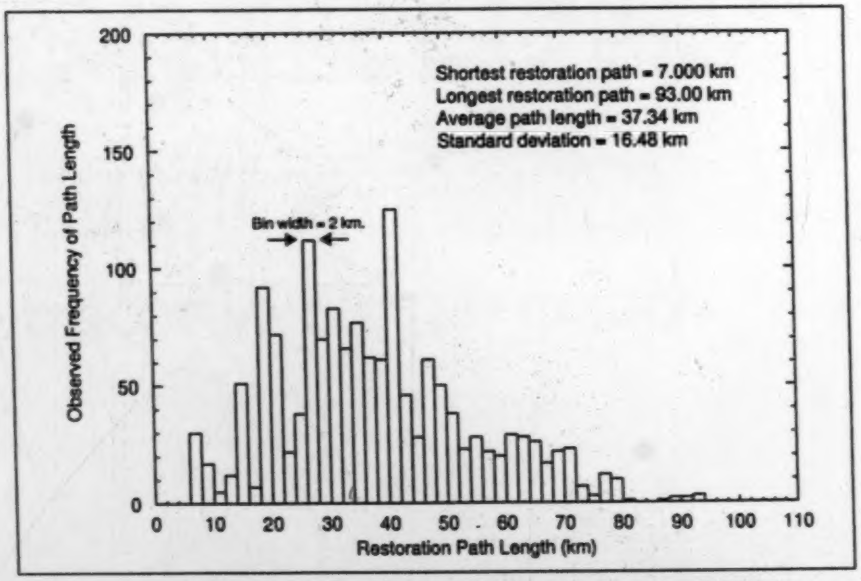


Figure 13.4. Distribution of restoration path lengths found by OPRA in a metropolitan network with stub release over all span cuts

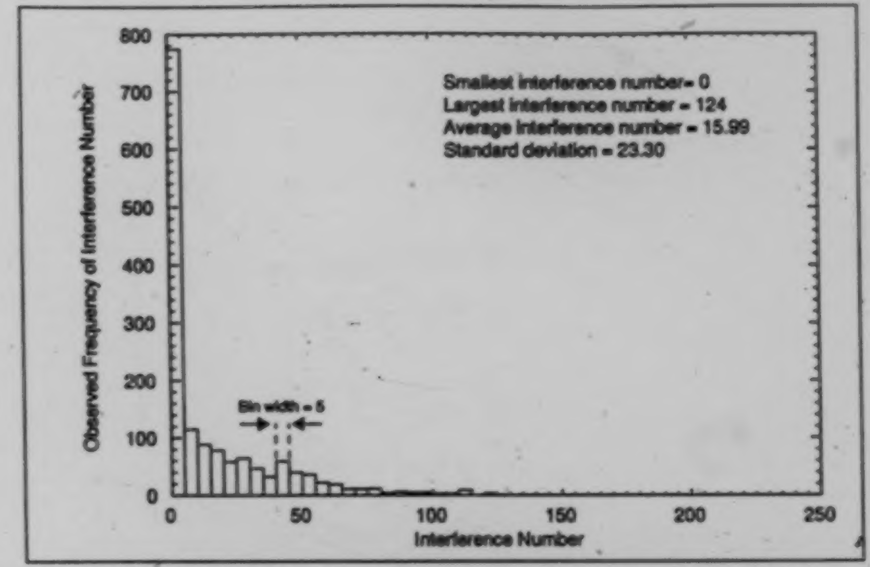


Figure 13.5. Distribution of interference numbers in a metropolitan network with stub release over all span cuts

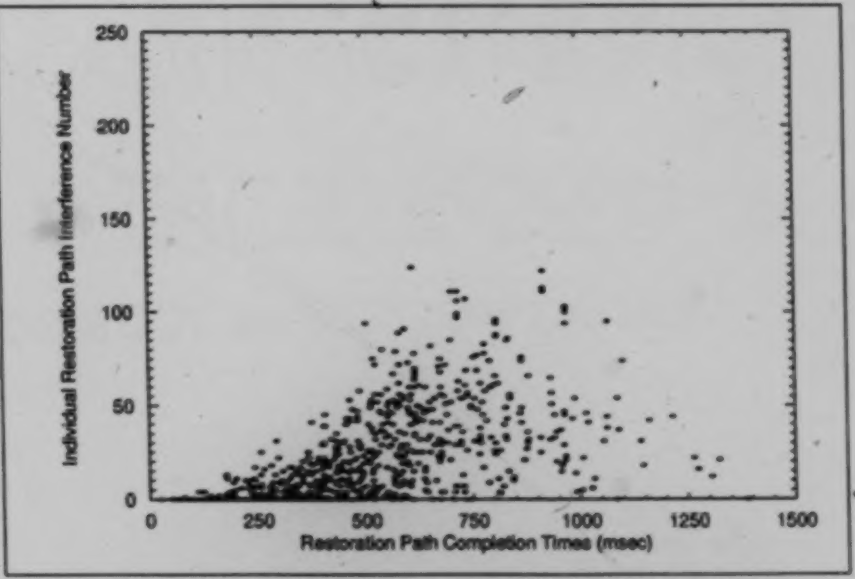


Figure 13.6. Interference number versus path time for all span cuts in a metropolitan network with stub release

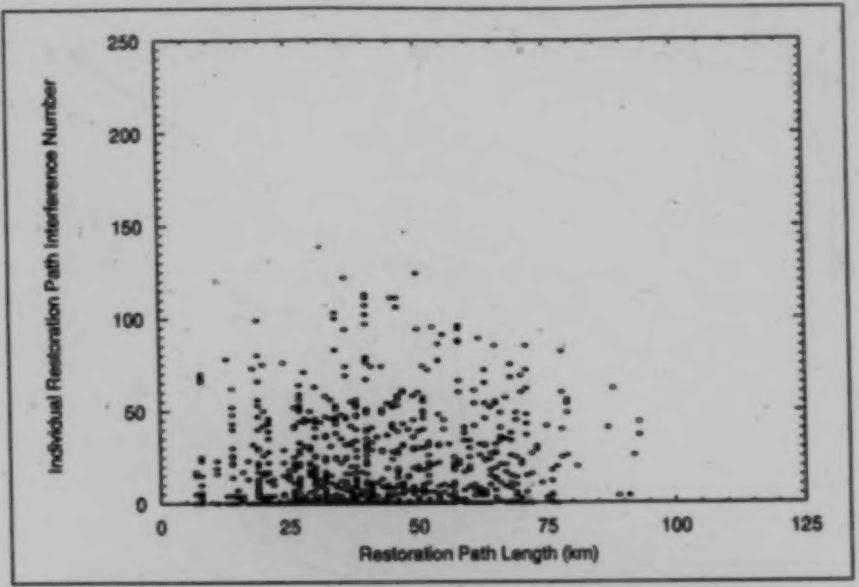


Figure 13.7. Interference number versus path length for all span cuts in a metropolitan network with stub release

## 13.7 Interpretation of Results

### 13.7.1 Working Path Failure Times

The working path failure times presented in Figure 13.1 approximate a Gaussian distribution with a standard deviation of 96.87 msec about an average of 209.9 msec. This histogram presents the times at which the nodes terminating a failed working path receive path level alarms. When the time scale of Figure 13.1 is compared to the time scale of the restoration trajectory plot shown in Figure 13.2, it is apparent that both the network failure and restoration process are active simultaneously.

### 13.7.2 Restoration Trajectories

Comparing the restoration trajectories presented in Figure 13.2 to those of the same network in Chapter 11 in Figure 11.8, it is evident that staggered alarms do not significantly reduce a network's restorability or increase a network's restoration time. The difference in network restorability, Rn, and average restoration time, $t_{p,avg}$, between the restoration trajectories shown in Figures 13.2 and 11.8 is only 0.2% and 21.3 msec respectively. Even with staggered alarms, a complete restoration plan is identified for the metropolitan network with stub release before the two second call dropping threshold.

While the average restoration times are very close, the restoration time of the first span to be fully restored, highlighted in red in both plots, is significantly less in Figure 11.8 when compared to Figure 13.2. Under staggered alarm conditions, span 0-14 is fully restored at a time of 353.9 msec, as opposed to 66.00 msec given a guillotine model of a cable cut. In general, the restoration trajectories of those spans restored first in Figure 11.8 follow a more gentle slope under staggered alarm conditions, and complete their restoration plan at a later time. However, the restoration time of those spans restored last in Figures 13.2 and 11.8 does not change significantly. In both plots the time required to find 0.95 of all restoration paths is close to 1 sec. Under staggered alarm conditions the skew in the restoration trajectories is less because restoration paths are found one at a time as alarms accumulate, instead of in bundles that drive the restorability of smaller spans to 100% very quickly.

### 13.7.3 Restoration Path Times

Comparing histograms 13.3 and 11.9 it apparent that staggered alarms tend to even out the distribution of restoration path completion times. Because restoration paths are found one at a time in Figure 13.3 rather than in bundles at discrete intervals, the probability that a large number of restoration paths are completed in a given 50 msec window is less. Compared to Figure 11.9, the histogram in Figure 13.3 lacks the large observed frequency of path times centred around 250 msec, and has a slightly larger $t_{p,avg}$, $t_{95}$, and standard deviation.

### 13.7.4 Restoration Path Lengths

Comparing histograms 13.4 and 11.10 it is evident that staggered alarms increase the average restoration path length slightly. The average path length in Figure 13.4 is 37.34 km, compared to 34.74 km in Figure 11.10. However, staggered alarms do not lead to individual restoration paths much longer than the average given that the length of the longest restoration path in Figure 13.4 is less than the length of the longest restoration path in Figure 11.8 by 15 km.

### 13.7.5 Interference Numbers

Comparing Figure 13.5 and 11.12 it is apparent that staggered alarms have a significant effect on the number of statelet families competing to restore lost capacity. Staggered alarms reduce the number demand pairs actively seeking restoration paths simultaneously. Consequently the value of the largest interference number, average interference number, and standard deviation in Figure 13.5 is approximately half of those values associated with a guillotine model of a cable cut.

### 13.7.6 Scatterplots

Comparing Figures 13.6 and 13.5 with Figures 11.13 and 11.14 respectively, it is apparent that staggered alarms eliminate large path interference numbers. Staggered alarms reduce the number of statelets seeking restoration paths simultaneously resulting in smaller interference numbers. The scatterplots are similar in all other respects. Those restoration paths with the lowest interference number are still usually restored first, and there is no discernible correlation between a restoration path's length and its interference number.

## 13.8 Conclusion

In summary, direct exposure of OPRA to staggered failure scenarios has little impact on $t_R$ and Rn. The restoration plan for each span failure in the metropolitan network with stub release is identified before the two second call dropping threshold just as with the guillotine model of a cable cut. The only significant difference between the results presented here and those presented in chapter 11 is a reduction in the number of restoration paths with large interference numbers.

# Chapter 14. Eliminating the Interference Heuristic In OPRA

This chapter quantifies the benefits of the interference heuristic which enables OPRA to synthesize a near optimal multi-commodity max-flow restoration pathset in usually less then two seconds. This pathset is optimal in the sense that it uses a near minimum amount of spare capacity when compared to the ideal IP reference solution. By eliminating the interference heuristic in OPRA, and satisfying the broadcast mesh of all statelets to the greatest extent possible in the order in which statelets arrive at a node, the benefits of synthesizing a restoration pathset using the interference heuristic can be quantified.

## 14.1 Eliminating the Interference Heuristic

The interference heuristic preferentially satisfies those incoming statelets seeking rebroadcast with the lowest interference number first. Those statelets with the lowest interference number do not traverse nodes of low degree, or spans of low sparing. Satisfying such statelets first was found to be beneficial based on experiments performed using the Interference Tester presented in chapter 6. The interference heuristic establishes a broadcast pattern which satisfies all statelets seeking rebroadcast at a node to the greatest extent possible based on a statelet's overall rank in terms of its interference number, and the relative spans in which it and other statelets lie. Late arriving statelets with low interference numbers are allowed to supplant outgoing statelets from other families which arrived earlier. Supplanting such high interference number statelets during the forward flooding process usually halts the reverse linking process of those statelets with higher interference numbers, and allows OPRA to resolve itself into those restoration paths with the lowest interference number.

Eliminating the interference heuristic in OPRA means ignoring a statelet's interference number. In this case the broadcast pattern of the statelet which arrived first at a tandem node, and is seeking rebroadcast, rather then that statelet with the lowest interference number, is always satisfied to greatest extent possible, consistent with the relative spans in which it and other statelets lie. Late arriving statelets with low interference numbers are *not* allowed to supplant outgoing statelets from other family's such that the broadcast pattern of the statelet which arrived first at a tandem node persists until collapsed during reverse linking or the disappearance of that statelet's precursor. Eliminating the interference heuristic transforms OPRA into a bidirectional flooding DRA which does not attempt to use a minimum amount of spare capacity.

## 14.2 Study Network

The interference heuristic affects a statelet's broadcast mesh the most when a network' s sparing is low, and the competition between statelets for outgoing spare links at a node is high. In this case the average interference number of a statelet is high. The benefits of the interference heuristic are most readily apparent in a network where the contention amongst statelets attempting to restore a span cut is high. Of all the networks investigated in chapter 11, those statelets from the metropolitan network without stub release had the highest average interference number. Therefore, the network used to quantify the benefits of the interference heuristic is the metropolitan network without stub release investigated in chapter 11.

## 14.3 Test Result Presentation Format

The plot of the restoration trajectories, along with the histograms of restoration path time and length, are presented here for the metropolitan network without stub release. Plots and histograms similar to those presented in chapter 11 concerning interference numbers are omitted here because they are not relevant.

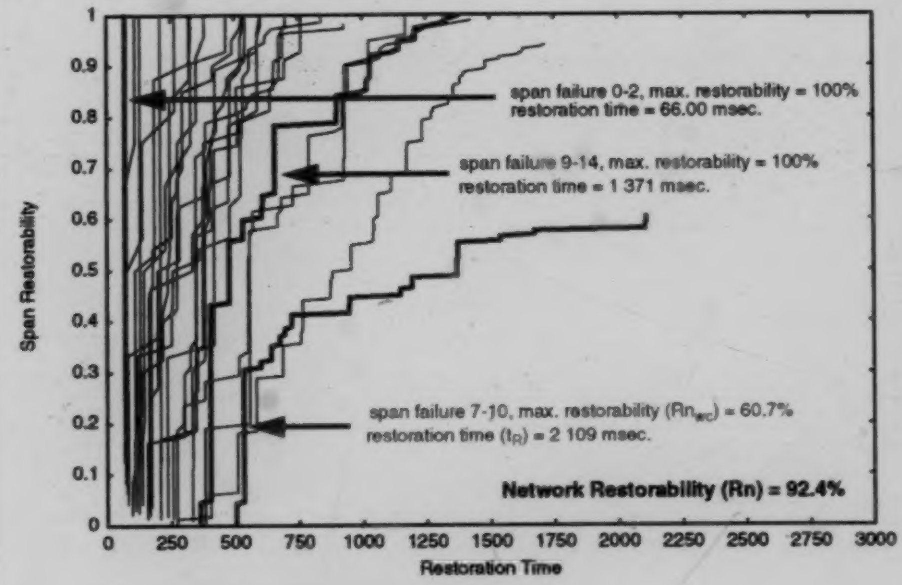
## 14.4 Option Settings

The same values assigned in chapter 11 to the various options defined in OPRA and the testbed are reused here. In summary they are:

IIN_STEP = 0
MAXREPEATS = 6
TIMEOUT = 3 sec.
transmission delay = 1.25 msec.
propagation delay calculated assuming FOTS
STATELET_PROCESSING_DELAY, = 1 msec.
BROADCAST_DELAY = 2 msec.
DELTA_T = 2 msec.

## 14.5 Experimental Results in a Metropolitan Network without Stub Release

The following plots display the performance of a bidirectional flooding path DRA which does not use the interference heuristic in a metropolitan network (network number 2), whose spare capacity is optimized to facilitate path restoration without stub release (design case number 2). This DRA is equivalent to OPRA without the interference heuristic. All results assume a guillotine model of a cable cut.



**Note:**

Time first alarm received at a node = 0.000 msec.
Time last alarm received at a node = 0.000 msec.

Time first restoration path found ($t_{p1}$) = 66.00 msec.
Time last restoration path found ($t_R$) = 2 109 msec.
Average restoration time ($t_{p,avg}$) = 546.2 msec.
Standard deviation from average restoration time = 349.7 msec.
Time required to find 0.95 of all restoration paths ($t_{95}$) = 1 334 msec.

Figure 14.1. Restoration trajectories in a metropolitan network without stub release

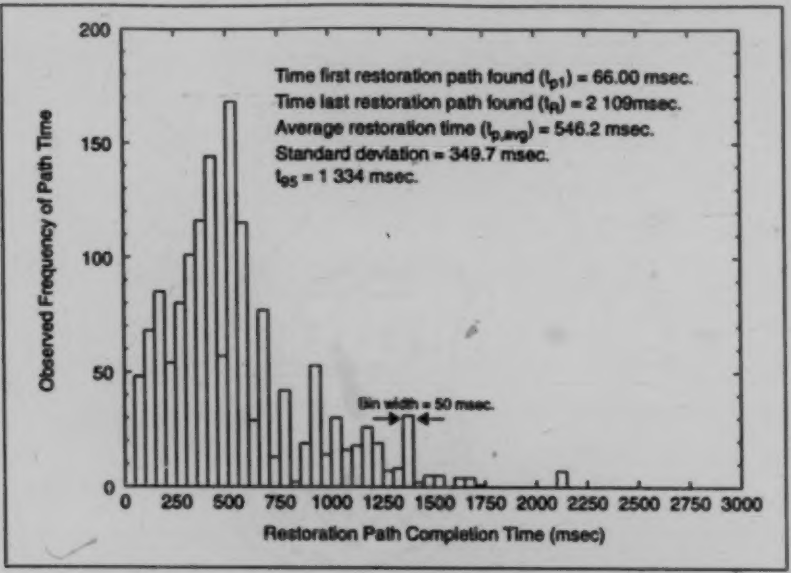


Figure 14.2. Distribution of restoration path times in a metropolitan network without stub release over all span cuts

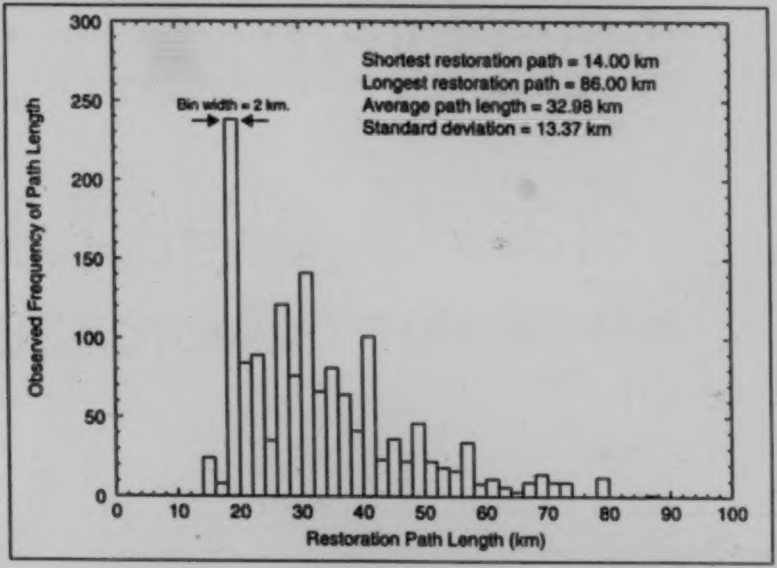


Figure 14.3. Distribution of restoration path lengths found by OPRA in a metropolitan network without stub release over all span cuts

## 14.6 Interpretation of Results

The benefits of the interference heuristic may be quantified by comparing the results presented here of a a bidirectional flooding path DRA which does not employ the interference heuristic to a bidirectional flooding path DRA which uses the interference heuristic, i.e. OPRA. These benefits are quantified below in terms of network restorability, Rn, and restoration time, i.e. $t_{p1}$, $t_R$, $t_{p,avg}$, and $t_{95}$.

### 14.6.1 Restoration Trajectories

Comparing the restoration trajectories presented in Figure 14.1 to those of the same network in chapter 11 in Figure 11.1, it is apparent that eliminating the interference heuristic in OPRA reduces the Rn by 6%. The benefits of the interference heuristic may therefore be quantified as 6% in terms of network restorability. However, this measure alone does not accurately capture the importance of the interference heuristic in OPRA because it includes the restorability of all those spans in a network which have only a few working links, and are easy to restore given the abundance of spare capacity available to these spans. In any network whose sparing is designed to restore all single span failures, the spare capacity requirements of a few single large spans usually determine that network's spare capacity placement, and usually only when these large spans are severed are the benefits of the interference heuristic truly revealed.

. The benefits of the interference heuristic can be better quantified when the $Rn_{wc}$ of the plot shown in Figure 14.1 is compared to the $Rn_{wc}$ of Figure 11.1. In both figures the $Rn_{wc}$ is for span failure 7-10, which is the second largest span in the network with 250 working links. The difference between these worst case restorabilities is a better measure because it is evaluated when the contention amongst statelets attempting to restore a span cut is high, and the interference heuristic is needed most to synthesize a near optimal multi-commodity max-flow pathset. This difference is 32.1% for the metropolitan network without stub release. A difference of this magnitude is very significant and reveals the importance of the interference heuristic in OPRA.

### 14.6.2 Restoration Path Times

Comparing histograms 14.2 and 11.2 it is apparent that eliminating the interference heuristic reduces the $t_{p,avg}$, $t_R$, and $t_{95}$. The reduction in $t_{p,avg}$ is small at 109.4 msec; however, the reduction in $t_R$, and $t_{95}$, is significant at 859 msec and 1 001 msec respectively. This reduction though, is not the result of a substantially faster bidirectional flooding path DRA when the interference heuristic is eliminated, as confirmed by the fact that the $t_{p1}$ of both histograms in Figures 14.2 and 11.2 are the same, rather it is mainly the result of not synthesizing as complete a restoration pathset for those large spans that are difficult to restore. Given that the interference heuristic only begins to affect a statelet's broadcast mesh when a network's sparing is low, and the competition between statelets for outgoing spare links at a node is high, it is likely that when a network is generously spared, or when a small span is cut, OPRA will have restoration times comparable to those of a bidirectional flooding path DRA which does not use the interference heuristic.

### 14.6.3 Restoration Path Lengths

Comparing the histograms in Figures 14.3 and 11.3 it is apparent that eliminating the interference heuristic does not significantly alter the restoration path length distribution. The shortest restoration path in Figure 14.3 has the same length as the shortest restoration path in Figure 11.3. Similarly, the longest restoration path in Figure 14.3 has the same length as the longest restoration path in Figure 11.3. In addition the average restoration path length and the standard deviation from this average in both histograms are nearly identical. These similarities may be attributed to the fact that OPRA and a bidirectional flooding path DRA which does not use the interference heuristic find similar paths when there is no contention for spare capacity in a network after severing a small span with few working links. Only when there is contention for spare capacity does OPRA synthesizes a more complete restoration pathset, which has little impact on the restoration path length distribution because the number of spans in the metropolitan network for which there is little contention for spare capacity is larger than the number of spans for which there is significant contention for spare capacity.

## 14.7 Conclusion

The interference heuristic is needed in OPRA to synthesize a near optimal multicommodity max-flow restoration pathset. The benefits of the interference heuristic are most apparent when the large spans in a network are severed and there is a lot of contention for spare capacity. This benefit is most accurately quantified comparing the $Rn_{wc}$ of OPRA to a path DRA which does not employ the interference heuristic. The results presented in this chapter show that eliminating the interference heuristic reduces the restorability of the second largest span of the metropolitan network without stub release by 32.1%.

# Chapter 15. Achieving Restoration Levels Proportional to a Network's Pre-Failure Connectivity After Multiple Span Failures

When a network is not fully restorable, it is desirable to achieve an overall pattern of individual restoration levels on each relation that is proportional to that network's pre-failure connectivity. In the event of a multiple span or node failure, there likely will be a shortage of spare capacity in a network efficiently provisioned for span recovery, and OPRA will not be able to fully restore every affected demand pair. In this case it is important to obtain some non-zero apportionment to every demand, ideally pro-rated to the pre-failure levels [28]. This chapter investigates OPRA's ability to restore multiple span failures, and synthesize a restoration pathset which apportions to every affected demand pair an amount of spare capacity proportional to that network's pre-failure connectivity.

## 15.1 Network Recovery from Multiple Span Failures and Node Loss

Whether one span, multiple spans, or a node fails in a network, OPRA should only synthesize restoration paths for those affected demand pairs which remain topologically connected. Only those demand pairs which remain topologically connected can be restored by OPRA, or any other restoration mechanism. If OPRA is only informed of these demand pairs, the simultaneous failure of one or more spans, or the simultaneous loss of one or more nodes, appears the same. The number of connected demand pairs affected by a failure does not impact or alter OPRA's path synthesizing capability. Though a simultaneous multiple span failure will likely affect more demand pairs than a single span failure, and a node failure will activate OPRA in those demand pairs which had working paths traversing the failed node, OPRA is unable to distinguish one case from the other.

Networks are usually provisioned to survive all individual span cuts (where a span is defined as the collection of all transport signals between a pair of adjacent nodes). In such 100% span restorable networks, there may be a shortage of spare capacity for recovery from node loss or multiple span failures. When it is impossible to fully restore every demand pair affected by a failure, it is important to obtain some non-zero apportionment to every demand, ideally pro-rated to the pre-failure levels [28]. OPRA attempts to do this using the constant IIN_STEP. This chapter investigates whether increasing IIN_STEP helps achieve an overall pattern of individual restoration levels on each relation that is proportional to that network's pre-failure connectivity after the simultaneous failure of multiple large spans.

As explained in chapter 8, when a unit of lost capacity is restored at a node, the IIN of all subsequent statelets originating from that node is increased by IIN_STEP. Increasing the IIN of those statelets sourced by a node increases the likelihood that other demand pairs which have not restored as much lost capacity as this node will complete one of their restoration paths. In a network which is not 100% restorable, this helps achieve an overall pattern of individual restoration levels on each relation that is more proportional to that network's pre-failure connectivity as shown in the results presented in this chapter.

In this thesis, the effects of varying the constant IIN_STEP are investigated by simulating multiple simultaneous span failures, rather than simulating node failures, because a maximum number of relations for which at least one restoration path exists may be affected. When a node fails it is impossible for any restoration mechanism to restore those working paths which either originate or terminate at the failed node. However, when large spans of a network which don't divide that network into two or more unconnected parts are failed simultaneously, a maximum number of topologically connected demand pairs are affected, and a shortage of spare capacity is likely, in which case OPRA will be vigorously challenged to achieve an overall pattern of individual restoration levels on each relation that is proportional to that network's pre-failure connectivity.

## 15.2 Study Network

The effects of varying IIN_STEP could be investigated in any 100% span restorable network, such as those presented in chapter 11, during the simultaneous failure of multiple large spans because numerous demand pairs would be affected and a shortage of spare capacity would ensue. However, to ensure that a *severe* shortage of spare capacity ensues, that network in chapter 11 with the lowest redundancy was chosen here to investigate the effects of varying IIN_STEP.

As shown in Table 11.3, the metropolitan network with stub release has lowest redundancy. Two spans were failed in this network to obtain the results presented in this chapter: spans 9-10 and 5-7. These spans are two of the largest spans in the metropolitan network with stub release. Span 9-10 consists of 122 working links and span 5-7 consists of 106 working links. Combined, the simultaneous failure of these spans affect a total of 20 connected demand pairs and 228 units of demand, and a severe shortage of spare capacity ensues.

## 15.3 Test Result Presentation Format: Starplots

Starplots are used to show the impact of varying IIN_STEP. As explained in [28], a starplot is a way of representing multi-dimensional data. Here it shows the fraction of demand restored for each connected relation affected by a failure via the length of a radius, all relations being displayed at equal angles around the circle. Ideally the star plot is a perfect circle as shown in Figure 15.1.

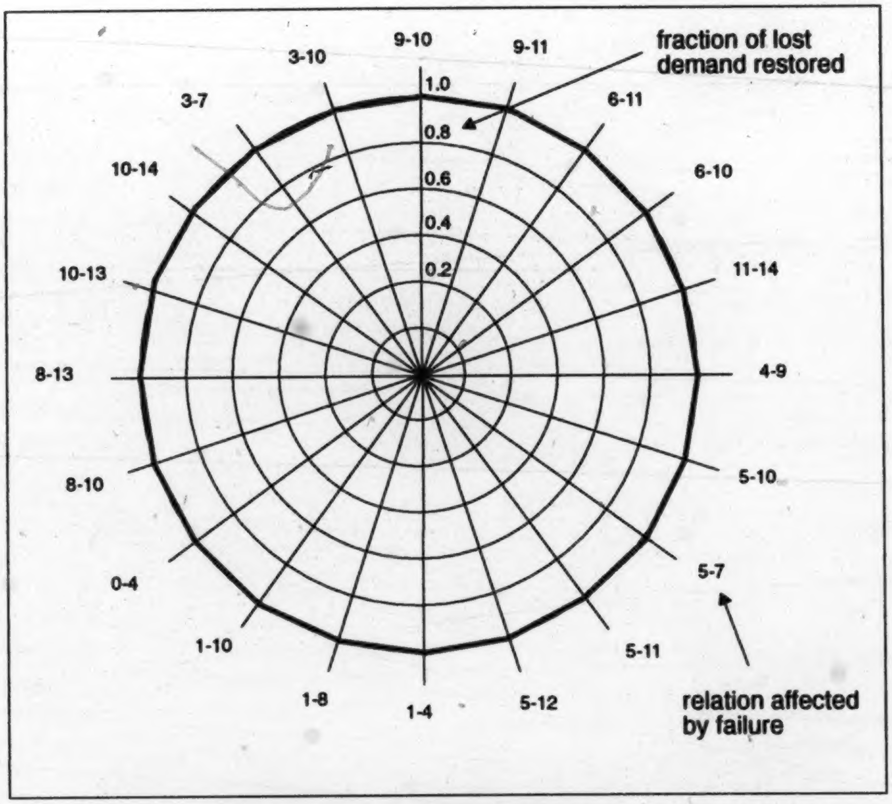


Figure 15.1. Ideal star plot for metropolitan network with stub release after the failure of spans 9-10 and 5-7.

## 15.4 Option Settings

To evaluate the impact of IIN_STEP on the overall pattern of individual restoration levels on each relation affected by the failure of spans 9-10, and 5-7, two sets of results are presented, one with IIN_STEP set to 0, and the other with IIN_STEP set to 50, which is close to the average interference number of a statelet in the metropolitan network with stub release as shown in Table 11.2. When IIN_STEP is set to zero, no relation is given preferential treatment while synthesizing restoration paths. When IIN_STEP is set to 50, OPRA favours restoring those relations which have restored as little as one unit of demand less than another relation because setting IIN_STEP close to the average interference number of a statelet will enable a node to distinguish those statelets which belong to a relation which has restored even one unit of demand more than another relation.

Other than the value assigned to IIN_STEP in the second set of results presented here, the same values assigned in chapter 11 to the various options defined in OPRA and the testbed are reused here. In summary they are:

MAXREPEATS = 6
TIMEOUT = 3 sec.
transmission delay = 1.25 msec.
propagation delay calculated assuming FOTS
STATELET_PROCESSING_DELAY, = 1 msec.
BROADCAST_DELAY = 2 msec.
DELTA_T = 2 msec.

## 15.5 Experimental Results in Metropolitan Network with Stub Release

The following starplots, and accompanying tables, show the impact of IIN_STEP on the overall pattern of individual restoration levels on each relation affected by the failure of spans 9-10, and 5-7, in the metropolitan network with stub release.

### 15.5.1 IIN_STEP Set to Zero

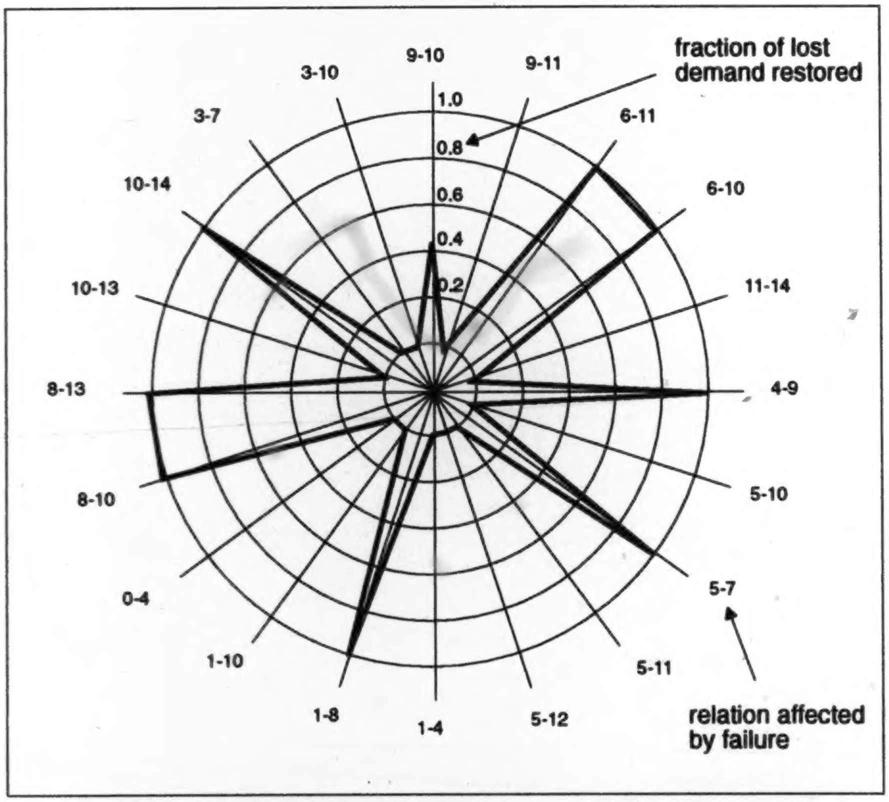


Figure 15.2. Star plot with IIN_STEP set to zero for metropolitan network with stub release after the failure of spans 9-10 and 5-7.

Table 15.1. Restored Capacity with IIN_STEP set to zero for metropolitan network with stub release after the failure of spans 9-10 and 5-7

| Affected demand pair | Lost demand | Restored demand | Restored demand normalized by lost demand |
|---|---|---|---|
| 4-9 | 12 | 12 | 1 |
| 5-10 | 52 | 0 | 0 |
| 5-7 | 8 | 8 | 1 |
| 5-11 | 4 | 0 | 0 |
| 5-12 | 4 | 0 | 0 |
| 1-4 | 4 | 0 | 0 |
| 1-8 | 1 | 1 | 1 |
| 1-10 | 4 | 0 | 0 |
| 0-4 | 4 | 0 | 0 |
| 8-10 | 6 | 6 | 1 |
| 8-13 | 1 | 1 | 1 |
| 10-13 | 17 | 0 | 0 |
| 10-14 | 25 | 25 | 1 |
| 3-7 | 4 | 0 | 0 |
| 3-10 | 1 | 0 | 0 |
| 9-10 | 40 | 18 | 0.45 |
| 9-11 | 2 | 0 | 0 |
| 6-11 | 4 | 4 | 1 |
| 6-10 | 31 | 31 | 1 |
| 11-14 | 4 | 0 | 0 |
| TOTALS | 228 | 106 | - |

### 15.5.2 IIN_STEP Set to Fifty

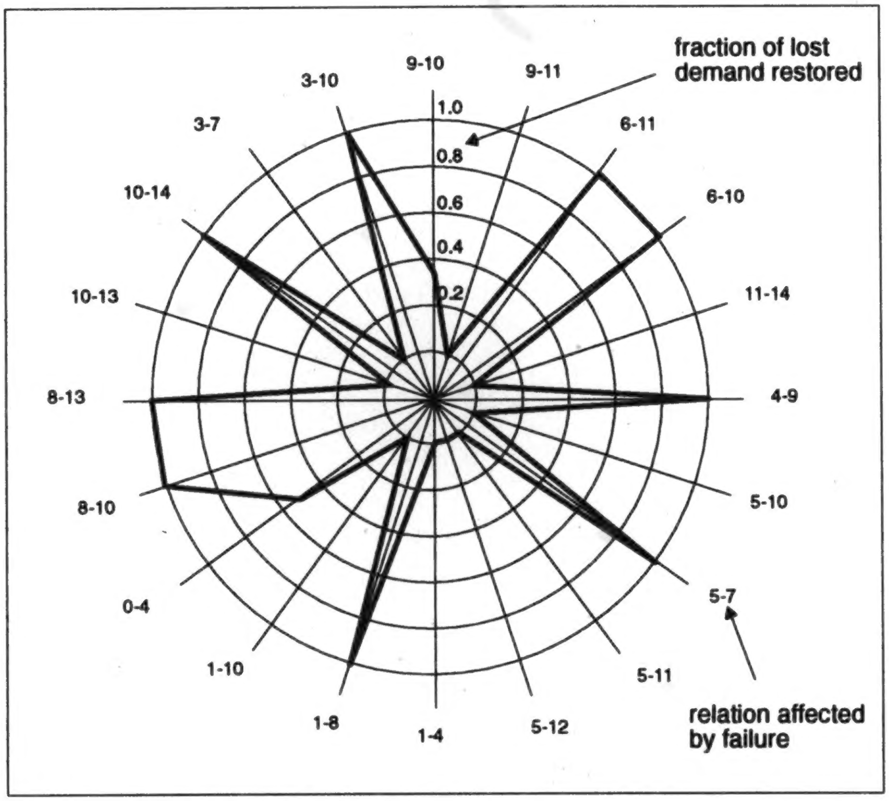


Figure 15.3. Star plot with IIN_STEP set to fifty for metropolitan network with stub release after the failure of spans 9-10 and 5-7.

Table 15.2. Restored Capacity with IIN_STEP set to zero for metropolitan network with stub release after the failure of spans 9-10 and 5-7

| Affected demand pair | Lost demand | Restored demand | Restored demand normalized by lost demand |
|---|---|---|---|
| 4-9 | 12 | 12 | 1 |
| 5-10 | 52 | 0 | 0 |
| 5-7 | 8 | 8 | 1 |
| 5-11 | 4 | 0 | 0 |
| 5-12 | 4 | 0 | 0 |
| 1-4 | 4 | 0 | 0 |
| 1-8 | 1 | 1 | 1 |
| 1-10 | 4 | 0 | 0 |
| 0-4 | 4 | 2 | 0.5 |
| 8-10 | 6 | 6 | 1 |
| 8-13 | 1 | 1 | 1 |
| 10-13 | 17 | 0 | 0 |
| 10-14 | 25 | 25 | 1 |
| 3-7 | 4 | 0 | 0 |
| 3-10 | 1 | 1 | 1 |
| 9-10 | 40 | 15 | 0.375 |
| 9-11 | 2 | 0 | 0 |
| 6-11 | 4 | 4 | 1 |
| 6-10 | 31 | 31 | 1 |
| 11-14 | 4 | 0 | 0 |
| TOTALS | 228 | 106 | - |

## 15.6 Interpretation and Discussion of Results

Ideally the post failure starplots shown in Figure 15.2 and 15.3 should be circles, like the starplot shown in Figure 15.1, with the exception that the radius of the post failure starplots may be less than one in the case of a multiple span failure in a network designed to survive all individual span failures, like the one investigated here. The significant disparity between the post failure starplots and the ideal circular starplot can be attributed to the fact that:

1. Speed as well as interference numbers moderate the competition between forward flooding statelets. Speed moderates competition between statelets in the sense that a quick forward flooding statelet with a large interference number may succeed at completing a restoration path if all ports part of that restoration path are locked, by setting the *activated* field of those ports to true, before another statelet with a smaller interference number supplants the high interference number statelet on those ports.
2. OPRA uses restoration paths one hop long whenever possible.
3. It may be topologically impossible to appoint to every affected demand pair a number of restoration paths proportional to that network's pre-failure connectivity through a failed network's surviving spare links.

Despite the non-ideal shape of the starplots shown in Figures 15.2 and 15.3, increasing IIN_STEP does help synthesize a restoration pathset which appoints to every affected demand pair an amount of capacity more proportional to that network's pre-failure connectivity, as shown in the results. Visually comparing these starplots it is apparent that increasing IIN_STEP has the desired effect of levelling the individual restorabilities on each relation. Comparing Tables 15.1 and 15.2, it is apparent that the PNE of relations 0-4, and 3-10, increase from zero to 50% and 100% respectively, and the PNE of relation 9-10 decreases from 45% to 35% when IIN_STEP is increased from zero to fifty. In both cases, 106 units of demand are restored. Increasing IIN_STEP from zero to 50 does not decrease the Rn of the metropolitan network without stub release.

In general, increasing IIN_STEP will not necessarily decrease the Rn of a network because this does not eliminate the interference heuristic within OPRA, rather it stratifies the competition between statelets such that only statelets from relations which have restored a similar amount of demand vie for available outgoing spare links at a node. All relations which have restored a similar amount of demand define a single band within which precursors compete for broadcast consistent with that precursor's overall rank in terms of the interference numbers of incoming statelets in that band and the spans in which it and other precursors lie. IIN_STEP determines how many bands exist and how clearly the boundaries between bands are marked. An IIN_STEP value of zero groups all statelets in a single band. A very large IIN_STEP value creates a band for each unit of demand restored and ensures statelets within different bands do not compete for broadcast at a node.

## 15.7 Conclusion

Increasing IIN_STEP helps synthesize a restoration pathset which appoints to every affected demand pair an amount of capacity proportional to that network's pre-failure connectivity. IIN_STEP is only needed when insufficient spare capacity exists to restore a network failure, which usually only occurs when two or more spans fail simultaneously or a node fails. IIN_STEP stratifies the competition between statelets such that only statelets from relations which have restored a similar amount of demand vie for available outgoing spare links at a node.

This chapter also shows that OPRA can restore single or multiple span and node failures. These failures may occur simultaneously or sequentially. OPRA is able to restore node failures because they appear the same as multiple span failures when OPRA is only informed of those affected demand pairs which remain topologically connected. OPRA is able to restore sequential failures which occur within the time it takes to restore one failure because such a failure scenario is equivalent to presenting OPRA with a set of staggered alarms, and OPRA is able to restore staggered alarms as shown in chapter 13. Therefore OPRA is capable of synthesizing a near optimal multi-commodity max-flow restoration pathset through a network's surviving spare links, independent of the distribution of alarms in time, as well as the number and location of connected demand pairs affected by a failure, as confirmed by the results presented in this and previous chapters.

# Chapter 16. Concluding Discussion

This chapter concludes this thesis with a review of the main development, a summary of the research contributions made, and recommendations for further research.

## 16.1 Review of Thesis

Chapter 1 explains the need for capacity efficient transport networks with short restoration times as a result of the widespread deployment of vulnerable high-capacity fiber optic transport facilities. Existing methods for transport network design and restoration are reviewed and distinguished from the research presented in this thesis. Through this comparison the potential benefits of path restorable mesh networks in terms of capacity efficiency are introduced. Two results are apparent at the end of chapter 1. The first is that the capacity placement of a path restorable mesh network may be efficiently determined using an IP formulation using flow constraints. The second result is that mesh restorable networks employing distributed dynamic path restoration have the greatest potential to restore a network failure quickly, require little administration overhead, handle numerous failure scenarios, are highly reliable, accommodate network growth, adapt themselves to any network topology, and require a minimum amount of spare capacity.

Chapter 2 provides background on the SONET signalling and control overheads that can be adapted to support state-based signalling. State-based signalling applies a set of static or quasi-static fields, called statelets here, repeatedly to individual links of a transport network. OPRA uses state-based signalling as the framework for communication. Chapter 2 also defines a reference architecture for the DCS machine capable of supporting the statelets OPRA interacts with.

Chapter 3 formally states the restoration problem using graph theoretical terms, and explains the difference between it and the packet routing and call routing problems. This chapter begins with a review of the technical terms defined in graph theory required to formally state the restoration problem and ends with a discussion of the complexity of the restoration problem. Chapter 3 concludes that the complexity of near optimal path restoration requires a distributed implementation to restore a failure within the two second call-dropping threshold in real time.

Chapter 4 describes a method for capacity optimization in path restorable networks, and quantifies the capacity benefits of path restoration over span restoration, as well as the further benefits of jointly optimizing working and spare capacity. This method uses flow constraints based on a suitable set of predefined routes over which pathsets must be implemented to formulate an integer program capable of optimizing the spare and/or working capacity of either a span or path restorable network with or without stub release. A completed IP run specifies not only the spare and working capacity per span, but also the corresponding routing of working and restoration paths. Though it is possible to use this restoration pathset information to achieve centralized restoration, it is used here instead to test OPRA's ability to efficiently restore a span cut. Chapter 4 concludes that mesh restorable networks using path restoration with stub release are the most capacity efficient. The findings presented in this chapter, along with the IP design method for path restorable networks, are one of the main contributions of this thesis.

Chapter 5 is devoted to developing appropriate figures of merit, by adapting those defined in [25] to a path DRA, through which the performance and efficiency of OPRA can be assessed systematically and quantitatively. The measures defined in chapter 5 may be categorized as operational performance metrics or intrinsic path metrics. The operational performance metrics include span and network restorability, $Rs_i$ and Rn, and speed related performance measures, $t_R$, $t_{95}$, $t_{p,avg}$, and $t_{p1}$. The intrinsic path metrics include path number efficiency (PNE) and path length efficiency (PLE). Both PNE and PLE are computed with respect to an ideal restoration pattern, and chapter 5 explains how the network designs presented in chapter 4 assist in this computation.

Chapter 6 presents the interference heuristic at the heart of OPRA through which the DRA self-organizes network spares into a near optimal multicommodity max-flow restoration pathset. The interference heuristic grew from the network level interference principle which recognized the advantages of avoiding the use of restoration paths which traverse spans with low sparing and/or nodes of low degree. Transforming this network level view of the interference principle into a distributed implementation requires associating an interference number with a statelet, and establishing a broadcast pattern locally at a node based on the interference number of incoming statelets. The broadcast pattern at a node is built up by forwarding that statelet with the lowest interference number on one link on all spans, except that span containing the incoming statelet, and satisfying the target broadcast pattern of all other incoming statelets to the greatest extent possible, consistent with a statelet's overall rank in terms of the interference numbers of incoming statelets and the spans in which it and other statelets lie. Every time a statelet is transmitted, its interference number is increased by the value of the interference number of the span it traverses. The interference number of a span is calculated locally at a node by counting the number of statelets competing to be forwarded in a span, and subtracting from this sum the number of spare links available for restoration in that span. The interference heuristic is a significant research contribution. Chapter 6 ends with the observation that a bidirectional flooding scheme is necessary to optimize the restoration pathset because it is an effective way to avoid the end-node bottleneck problem. Spans local to an end node of a demand pair will by this principle be quickly seized into anchoring paths for that demand pair. Recognizing and solving the end-node bottleneck problem also is a significant research contribution.

Chapter 7 explains distributed interaction via statelets and the logical environment at a DCS within which OPRA functions. This method of interaction stems from work done by W. D. Grover [25]. During the course of this explanation, the fields of a statelet are defined and their content and uses introduced. Among the advantages of using statelets is the result that OPRA effectively executes in an active memory space that is continually updated in a manner that implicitly reflects the properties of the host network, even though no direct knowledge of that network is stored by OPRA. Chapter 7 also introduces the set of DCS port registers which comprise the active memory space within which OPRA executes, and through which it indirectly interacts with the world outside its node.

Chapter 8 presents a detailed description of OPRA, emphasizing the logic necessary to analyse the various events encountered during the restoration process, and thereby determine the correct processing of statelets. The logic which forms OPRA is encoded in an event-driven FSM. When an instance of OPRA is executed concurrently and asynchronously at every node of a network, responding solely to statelets, with no direct interaction with other nodes and with no stored knowledge of network topology, each node will derive isolated crosspoint operate decisions which collectively synthesize a near optimal restoration pathset between the source and destination of all demand pairs affected by a failure when viewed at the network level. OPRA represents the second main contribution of this thesis.

Chapter 9 presents the procedures required to effectively test OPRA using discrete event simulations using the polling implementation detailed in chapter 10. The procedures presented in this chapter are needed to exploit the implementation technique used to test OPRA in this thesis.

Chapter 10 describes the testbed in which OPRA was implemented for experimental characterization. OPRA is a transient, non-linear, massively parallel, multi-dimensional process. For these reasons an experimental research method was adopted in which OPRA was actually implemented and executed in every node of the various study networks using concurrent programming methods. This research methodology essentially constitutes a series of experiments on a prototype implementation as opposed to results obtained by simulation of the intended process.

The remaining chapters in this thesis analyse OPRA's performance using the testbed explained in the last chapter. Chapter 11 evaluates OPRA's ability to restore all single span failures in a metropolitan and a long haul network. OPRA restores all single span failures in a network quickly and efficiently, and given the conservative processing delays and tightly designed networks used to find the results presented in this chapter, would likely complete a restoration plan which fully restores any single span failure before the two second call dropping threshold in each of these networks. Furthermore, the consistency of OPRA's restoration time and PNE in all of the tests performed, and summarized in chapter 11, demonstrate the algorithm's flexibility and insensitivity to network size and topology.

Chapter 12 investigates the impact of reducing the processing delay associated with a statelet. The processing delays considered in this chapter are less than those used in chapter 11 but remain realistic and comparable to the values chosen previously [21]. Chapter 12 shows that reducing the processing delay reduces the restoration time without degrading a network's restorability, and verifies that it is likely a complete restoration plan which fully restores any single span failure would be identified by OPRA before the two second call dropping threshold in any network.

Chapter 13 reports the behaviour of OPRA when presented with a random series of individual link failures, as opposed to a "guillotine" model of a cable cut, and verifies OPRA's ability to cope with complex failure stimuli. This chapter shows direct exposure of OPRA to staggered failure scenarios has little impact on $t_R$ and Rn. The only significant difference between random and simultaneous link failures is a reduction in the number of restoration paths with large interference numbers. This is because staggered alarms reduce the number demand pairs actively seeking restoration paths simultaneously.

Chapter 14 quantifies the benefits of the interference heuristic by eliminating the interference heuristic in OPRA and satisfying the broadcast mesh of all statelets to the greatest extent possible in the order in which statelets arrive at a node. The benefits of the interference heuristic are most apparent when the large spans in a network are severed and there is significant contention for spare capacity. This benefit is most accurately quantified comparing the $Rn_{wc}$ of OPRA to a path DRA which does not employ the interference heuristic. The results presented in this chapter show that eliminating the interference heuristic can reduce the restorability of a large span by up to 32%.

Chapter 15 investigates OPRA's ability to restore multiple span failures, and synthesize a restoration pathset which appoints to every affected demand pair an amount of spare capacity proportional to that network's pre-failure connectivity. This chapter shows that IIN_STEP helps achieve this goal by stratifying the competition between statelets such that only statelets from relations which have restored a similar amount of demand vie for available outgoing spare links at a node. In addition, this chapter in conjunction with chapters 11 through 14 confirms that OPRA is capable of synthesizing a near optimal multi-commodity max-flow restoration pathset through a network's surviving spare links, independent of the distribution of alarms in time, and independent of the number and location of connected demand pairs affected by a failure.

## 16.2 Summary of Research Results

The main contributions of this thesis are:

1. a path restorable network capacity design methodology using integer programming;
2. an optimized, distributed, real-time solution to the path restoration problem, named OPRA;
3. the interference heuristic which at the node level, recognizes the benefits of satisfying the target broadcast pattern of all incoming statelets based on their interference number from lowest to highest, where a transmitted statelet's interference number is the sum of the interference numbers of all spans the statelet traverses; and
4. the end-node bottleneck traversal problem, which in a path restorable network recognizes the benefits of quickly seizing the spans local to an end node of a demand pair in order to anchor paths for that demand pair.

Together these advances make possible path restorable transport networks which restore failures within two seconds, require little administration overhead, handle numerous failure scenarios, are highly reliable, easily accommodate network growth, adapt themselves to any network topology, and require a minimum amount of spare capacity. In the envisaged network, capacity would be minimized using the IP design methodology presented in chapter 4, and failures restored in real time in an entirely autonomous distributed manner using OPRA, or any other path DRA similar to OPRA which employs the interference heuristic and solves the end-node bottleneck traversal problem.

In addition, the following aspects of this thesis, although used to assist in the development of OPRA, are original and merit mentioning as research contributions because of their alternate uses as explained below:

1. the conventional program written to test the interference heuristic named Interference Tester. This program identifies a restoration pathset that could be used to achieve centralized restoration.
2. the optimal routing of restoration paths as identified by the IP used to evaluate the performance of OPRA. This pathset could also be used to achieve centralized restoration.
3. the OPNET testbed in which OPRA was implemented for experimental characterization. This testbed could be used to test many other path or span DRAs.

## 16.3 Further Research

### 16.3.1 Implementing the Interference Heuristic In ATM Networks

The interference heuristic has been explained assuming an STM technology platform like SONET. SONET will likely be the physical layer of future ATM networks, and making the physical layer of a transport network responsible for the restoration of a failure is often beneficial. However, several intrinsic features associated with ATM could potentially be exploited to provide improved restoration techniques beyond those established for SONET. The most important of these features are: ATM cell level error detection, inherent rate adaptation, and nonhierarchial multiplexing. Regardless of the technology platform made responsible for network restoration, if a network failure is restored using distributed path restoration, the interference heuristic can guide the algorithm such that capacity efficient replacement paths are found. How the interference heuristic is adapted to find a restoration pathset in an ATM environment is briefly discussed next.

When calculating the interference number of a span in an STM-based network as shown in Figure 6.3 in chapter 6, the spare capacity of a span is determined by counting the number of spare links available on the span. In ATM the spare capacity of a span cannot be determined by counting the number of links available on a span. Instead, the total spare capacity available on a span is determined by the sum of the spare capacity available on each of the span's constituent links, which may be some fraction of their total capacity.

In STM networks the bandwidth of each digital signal is fixed, and the unit of bandwidth a restoration statelet signal is seeking to restore is implicitly determined by the bandwidth of the link on which the signal is transmitted. Thus in an STM network, restoration statelet signals do not need to explicitly state the amount of capacity they are seeking to restore. In ATM the bandwidth of a working path is not fixed; the route and bandwidth of a digital signal are disassociated. In this case, a restoration statelet must specify the amount of capacity it is seeking to restore.

The decoupling of a restoration path's bandwidth and its routing in an ATM network generalizes the calculation of a span's interference number. The interference number of a span in an ATM network is calculated by subtracting the sum of the capacities specified by all restoration statelet signals waiting to access a span from the total sparing on that span. Whereas the interference number of a span in an STM network will always be an integer, in an ATM network the interference number of a span may be a real number.

ATM networks are still in their infancy relative to SONET networks, and the debate about which layer (either the ATM or SONET layer) of a transport network should be used to restore a failure is currently unresolved. Given these circumstances the description of the logic necessary for determining the correct response by OPRA to various restoration events assumed a SONET technology platform in this thesis. The description of the logic necessary for determining the correct response by OPRA to various restoration events assuming an ATM technology platform requires further research.

### 16.3.2 Further Studies In Capacity Placement

The capacity design methodology presented in this thesis applies directly to SHN networks like SONET. Extending the IP formulation presented in chapter 4 to ATM network requires further research. Without engineering a suitable bandwidth margin on all links in an ATM network sufficient to carry the traffic from a combination of working and backup VP routes, the GOS of restored VPs cannot be accurately predicted.

The sensitivity of the of the capacity placement results presented in chapter 4 to the size of the candidate route set chosen for evaluation by the IP should also be investigated. This would require comparing the total link count of the same network for different candidate route set size.

Finally the impact of modularity on the capacity design of a network should be investigated. In practice, links are provisioned in modules of a certain size, and before a new module is added, the total capacity requirements of a network may be minimized if the unused capacity in existing modules is used to form a new path.

### 16.3.3 Effects of Network Topology on Capacity Placement In Mesh Restorable Networks

The topology of a network will affect how much more capacity efficient a path restorable design is than a span restorable design, and how much a path restorable design will benefit from stub release. If most of the demand in a network is between the immediate end-nodes of a span, then in the limit, path restoration would not be much more capacity efficient than span restoration. On the other hand, when the source and destination nodes of demands affected by a span failure are separated from each other by many hops, span and path restoration will result in very different restoration pathsets, and path restoration will likely show significantly better capacity efficiency. The effects of network topology on capacity placement in both span and path mesh restorable networks requires further research.

### 16.3.4 Multi-Commodity Max-Flow Pathset Characterization

In order to solve the capacity placement problem in a path restorable network using heuristics as explained in chapter 1, it is beneficial to have a fast centralized ksp-like algorithm which identifies the restoration pathset and the multicommodity max-flow through the spare capacity of a path restorable network between those relations affected by a failure. Finding such an algorithm which characterizes a multi-commodity max-flow pathset requires further research.

### 16.3.5 A New Approach to Avoiding the End-Node Bottleneck Traversal Problem

Bidirectional flooding is used in OPRA to avoid the end-node bottleneck traversal problem. When a path DRA employs bidirectional flooding, a complemented statelet can be over-written at the source of the forward flooding statelet with which it is matched, as shown in the illustrative example presented in Figure 16.1. Only after a DRA employing bidirectional flooding completes a confirmation process similar to the one described in chapter 8, is a source/destination guaranteed that an incoming complemented statelet acts as a bi-directional holding thread between itself and the source of that statelet.

If ordinary flooding as shown in Figure 6.5 in chapter 6 is used by a path DRA instead of bidirectional flooding, a complemented statelet arriving at its destination node identifies a restoration path that will persist until those statelets that act as a bi-directional holding thread for that path are cancelled. However, this does not mean reverse linking would never be suspended in an ordinary flooding path DRA employing the interference heuristic, because precursors may still be overwritten at tandem nodes during the forward flooding process by late arriving statelets from other families with lower interference numbers. Ordinary flooding would only ensure that a complemented statelet which arrives at its destination node would not disappear, eliminating the need for a confirmation phase similar to the one required in OPRA. However, if bidirectional flooding is not used by a path DRA, another way must be found to avoid the end-node bottleneck traversal problem. Identifying a path DRA which does not use bidirectional flooding to avoid the end-node bottleneck traversal problem is a subject of further research.

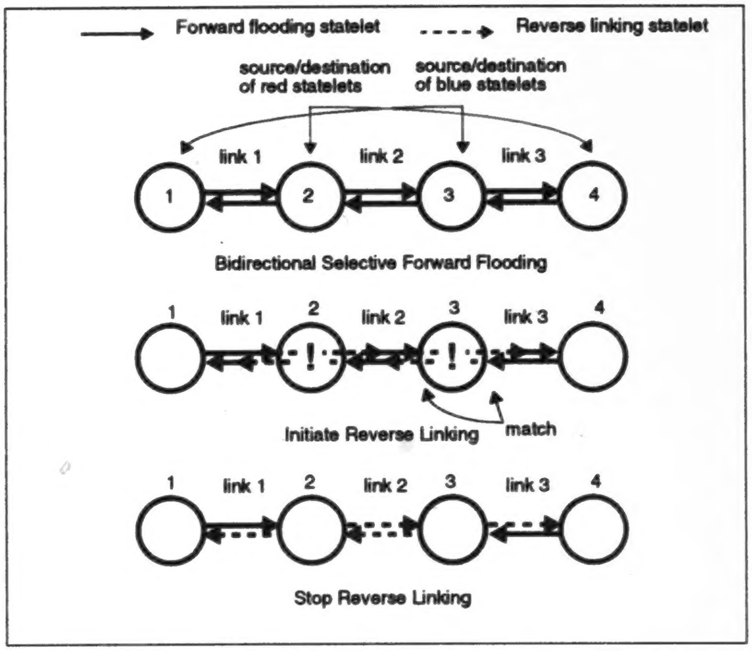

Forward flooding statelet
Reverse linking statelet
source/destination of red statelets
source/destination of blue statelets
link 1
link 2
link 3
1
2
3
4
Bidirectional Selective Forward Flooding
!
Initiate Reverse Linking
match
Stop Reverse Linking

Figure 16.1. Suspending reverse linking as a result of bidirectional flooding

### 16.3.6 Alarm Distribution in the Event of Node or Multiple Simultaneous Span Failures

Whether multiple spans, or a node fails in a network, OPRA should only synthesize restoration paths for those affected demand pairs which remain topologically connected. Only those demand pairs which remain topologically connected can be restored by OPRA, or any other restoration mechanism. In the case of a node failure, or a multiple simultaneous span failure which divides a network into two or more disconnected parts, alarm information should be distributed such that OPRA is only activated in those demand pairs which remain topologically connected. For example, in the case of a node failure, this means activating OPRA only in those demand pairs which had working paths traversing the failed node. Alarm distribution in the event of node failures or multiple span failures is a subject of further research.

### 16.3.7 Reversion

Reversion refers to the process by which a network is returned to its original state after a failure has been repaired. Reverting to the prefailure network configuration is made more difficult with stub release because a restoration path may use a released working link to restore a failure. In this case reversion would require releasing the restoration path and reconfiguring the working path, instead of simply moving the restored transport signal from the restoration path to the repaired working path. If reversion is impractical, it may be desirable to limit the use of spares obtained through the process of stub release to that demand pair which originally used that working path. Whether reversion is needed at all in a path restorable network using stub release is currently being investigated [42]. However, the spare capacity design of reversionless networks must consider not only spare capacity sharing but also route transitions [42]. Reversion remains a subject of further research in path restorable networks employing stub release.

### 16.3.8 Bypassing Tandem Nodes

When a path restorable network does not employ stub release, terminating working paths at tandem nodes is unnecessary. Furthermore, it is beneficial to bypass the DCS at a tandem node because the port count, and therefore the size of the DCS, at these nodes is reduced. Whether the savings resulting from bypassing a DCS is larger than the savings resulting from stub release is a subject of further research. If it is impractical to release all of the surviving portions of a cut working path, referred to as full stub release, it may be practical to release some, but not all, of the surviving links in a cut working path. This "partial" stub release also requires further research.

# References

[1] Anderson, J., Doshi, B.T., Dravida, S., Harshavardhana, P., "Fast Restoration of ATM Networks", *IEEE J-SAC Special Issue: Integrity of Public Telecommunication Networks*, vol. 12, no. 1, Jan. '94, pp. 128 - 138.

[2] Baker, J. E., "A distributed link restoration algorithm with robust preplanning", *Proc. IEEE GLOBECOM '91*, Dec. 1991, pp. 306-311.

[3] Bates, B., Gregory, D., Voice and Data Communications Handbook, New York, NY: McGraw-Hill Inc., 1996.

[4] Barezzani, M., Pedrinelli, E., Gerla, M., "Protection planning in transmission networks", *Proc. IEEE ICC'92*, 1992, pp. 316.4.1-316.4.5.

[5] Busche, M.T., Lockhart, C.M., Olszewski, C., "Dynamic k-shortest path (DKSP) facility restoration algorithm", *Proc. IEEE GlobeCom '94*, Dec. 1994, pp. 536 - 542.

[6] *Capacity Scavenging in the Telecom Canada Network: A Preliminary assessment*, Telecom Canada Report - TM 91.015 (CR 89-16-01).

[7] Chao, C. W., Dollard, P. M., Weythman, J. E., Nguyen, L. T., Eslambolchi, H., "FASTAR-a robust system for fast DS3 restoration", *Proc. IEEE GLOBECOM '91*, Dec. 1991, pp. 39.1.1-39.1.5.

[8] Chao, C.W., Fuoco, G., Kropfl, D., "FASTAR platform gives the network a competitive edge", *AT&T Technical Journal*, July/August 1994, pp. 69 - 81.

[9] Chng, R.S.K., Botham, C.P., Johnson, D., Brown, G.N., Sinclair, M.C., O'Mahony, M.J., Hawker, I., "A multi-layer restoration strategy for reconfigurable networks", *Proc. IEEE GlobeCom '94*, Dec. 1994, pp. 1872 - 1878.

[10] Chow, C. E., Bicknell, J. D., Mccaughey, S., "Perfromance analysis of fast distributed link restoration algorithms", *Internationsl Journal of Communication Systems*, vol. 8, 1995, pp. 325 - 345.

[11] Chujo, T., Komine, H., Miyazaki, K., Ogura, T., Soejima, T., "Distributed self-healing network and its optimum spare capacity assignment algorithm", *Electronics and Communications in Japan*, part 1, vol. 74, no. 7, 1991, pp. 1-8.

[12] Coan, B.A., Vecchi, M. P., Wu, L.T., "A distributed protocol to improve the survivability of trunk networks", *Proceedings of the 13th International Switching Symposium*, May 1990, pp. 173 - 179.

[13] Coan, B.A., et al., "Using distributed topology updates and preplaned configurations to achieve trunk network survivability", *IEEE Transaction on Reliability*, vol. 40, no.4, 1991, pp. 404-416

[14] Davis, L., Cox, A., Qiu, Y., "A Genetic Algorithm for Survivable Network Design", *Proc. of the Fifth International Conference on Genetic Algorithms*, July, 1993, pp. 408 - 415.

[15] D'egidio, S., Reagor, B. T., "What do we do now?", *Bellcore exchange*, October 1994, pp. 4-9.

[16] *Digital Network Notes*, Technical Planning and Standards Engineer, Telecom Canada, 410 Laurier Ave. West, Ottawa, Ontario, Canada, K1P 6H5, 1984.

[17] Dijkstra, E.W., "A note on two problems in connection with graphs", *Numer Math.* vol. 1, 1959, pp. 269-271.

[18] Doverspike, R. D., Morgan, J. A., Leland, W., "Network design sensitivity studies for use of digital cross-connect systems in survivable network architectures", *IEEE J-SAC Special Issue: Integrity of Public Telecommunication Networks*, vol. 12, no. 1, Jan. '94, pp. 69 - 78.

[19] Dunn, D.A., Grover, W.D., MacGregor, M.H., "Comparison of k-shortest paths and maximum flow routing for network facility restoration", *IEEE J-SAC Special Issue: Integrity of Public Telecommunication Networks*, vol. 12, no. 1, Jan. '94, pp. 88-99.

[20] Freeman, R.L., *Telecommunication System Engineering, Second Edition*, N.Y., N.Y.: John Wiley and Sons Inc., 1989.

[21] Fujii, H., Yoshikai, N., "Restoration message transfer mechanism and restoration characteristics of double-search self-healing ATM network", *IEEE J-SAC Special Issue: Integrity of Public Telecommunication Networks*, vol. 12, no. 1, Jan. '94, pp. 149 - 158.

[22] Gardner, L.M., Heydari, M., Shah, J., Sudborough, I.H., Tollis, I.G., Xia. C., "Techniques for finding ring covers in survivable networks", *Proc. IEEE GlobeCom '94*, Dec. 1994, pp. 1862 - 1866.

[23] Gersht, A., Kheradpir, S.,"Real-time bandwidth allocation and path restoration in SONET based self-healing mesh networks", *Proc. IEEE ICC '93*, May 1993.

[24] Gibbons, A., *Algorithmic Graph Theory*, Cambridge University Press, 1985.

[25] Grover, W.D., *Selfhealing Networks - A Distributed Algorithm for k-shortest link-disjoint paths in a multi-graph with applications in realtime network restoration*, Ph.D. Dissertation, University of Alberta, Fall, 1989.

[26] Grover, W.D., "A frame-bit modulation technique for addition of tranparent signalling capacity to the DS3 signal", *Proc IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, June 1987.

[27] Grover, W.D., "Case studies of survivable ring, mesh, and mesh-arc hybrid networks", *Proc. IEEE Globecom '92*, Dec. 1992, pp. 633-638.

[28] Grover, W.D., "Distributed Restoration of the Transport Network", Chapter 11, pp. 337 - 419 of *Telecommunications Network Management into the 21st Century - Techniques, Standards, Technologies, and Applications*, edited by S. Aidarous and T. Plevyak, New York, NY: IEEE Press, 1994.

[29] Grover, W.D., "The Selfhealing network: a fast distributed restoration technique for networks using digital cross-connect machines", *Proc. IEEE Globecom '87*, Dec. 1987, pp. 28.2.1-28.2.6

[30] Grover, W.D., Slevinsky, J.B., MacGregor, M.H., "Optimized design of ring-based survivable networks", *Canadian Journal of Electronics and Computer Engineering*, Vol. 20, No. 3, 1995, pp.139 - 149.

[31] Grover, W., Venables, B.D., *The Selfhealing Network Protocol: Functional Description and Implementation*, TR-92-10, TRLabs Technical Report, 1992.

[32] Grover, W., Venables, B.D., "Performance of the Selfhealing Network protocol with random individual link failure times", *Proc. IEEE ICC '91*, 1991, pp. 22.2.1-22.2.7.

[33] Grover, W.D., Venables, B.D., MacGregor, M.H., Sandham, J. H., "Development and performance verification of a distributed asynchronous protocol for real-time network restoration", *IEEE J-SAC Special Issue: Computer-Aided Modelling, Analysis and Design in Communication Networks*, Vol. 9, No. 1, 1991, pp.112 - 125.

[34] Grover, W.D., Venables, B.D., Sandham, J. H., Milne, A. F., "Performance studies on a selfhealing network protocol in Telecom Canada long haul networks", *Proc IEEE Globecom '90*, San Diego, 1990, pp. 403.3.1 - 403.3.7.

[35] Hasegawa, S., Okanoue, Y., Egawa, T., Sakauchi, H., "Control algorithms of SONET integrated self-healing networks", *IEEE J-SAC Special Issue: Integrity of Public Telecommunication Networks*, vol. 12, no. 1, Jan. '94, pp. 110 - 119.

[36] Herzberg, M., Bye, S., "An optimal spare-capacity assignment model for survivable networks with hop limits", *Proc. IEEE GLOBECOM '94*, Dec. 1994, pp. 1601-1607.

[37] Hu, T.C., *Integer Programming and Network Flows*, Reading, MA: Addison-Wesley, 1969.

[38] Iraschko, R.R., "IP Network Designs", TRLabs Technical Report *TR-96-10*, October, 1996.

[39] Iraschko, R.R., "OPRA Testbed C-code Specification", TRLabs Technical Report *TR-96-11*, October, 1996.

[40] Iraschko, R.R., "OPRA C-code Specification", TRLabs Technical Report *TR-96-12*, October, 1996.

[41] Iraschko, R.R., MacGregor, M. H., Grover, W. D., "Optimal Capacity Placement for Path Restoration in Mesh Survivable Networks", *Proc. IEEE ICC'96*, June 1996.

[42] Kawmaura, R., Sato, K., Tokizawa, I., "Self-healing ATM networks based on virtual path concept", *IEEE J-SAC Special Issue: Integrity of Public Telecommunication Networks*, vol. 12, no. 1, Jan. '94, pp. 120 - 127.

[43] Komine, H., Chujo, T., Ogura, T., Miyazaki, K., Soejima, T., "A distributed restoration algorithm for multiple-link and node failures of transport networks", *Proc. IEEE Globecom '90*, Dec. 1990, pp. 459 - 463.

[44] Landegem, T. V., Vankwikelberge, P., Vanderstraeten, H., "A self-healing ATM network based on multilink principles", *IEEE J-SAC Special Issue: Integrity of Public Telecommunication Networks*, vol. 12, no. 1, Jan. '94, pp. 139 - 148.

[45] Lin, N.D., Zolfaghari, A., Lusignan, B., "ATM virtual path self-healing based on a new path restoration protocol", *Proc. IEEE GlobeCom '94*, Dec. 1994, pp. 794 - 798.

[46] MacGregor, M.H., *The Self Traffic-Engineering Network*, Ph.D. Dissertation, University of Alberta, Fall, 1991.

[47] MacGregor, M.H., Grover, W.D., "Investigation of a Cut-tree approach to network restoration from node loss", *Proc. IEEE ICC'95*, June 1995, pp. 1530-1535.

[48] MacGregor, M.H., Grover, W.D., "Optimized k-shortest paths algorithm for facility restoration", submitted to *IEEE Trans. Comm.*, Sept. 1992.

[49] MacGregor, M.H., Grover, W.D., Maydell, U. M., "The self-traffic-engineering network", *Canadian Journal of Electrical and Computer Engineering*, Vol. 18, No. 2, 1993, pp. 47 - 58.

[50] MacGregor, M.H., Grover, W.D., Ryhorchuk, K., "Optimal spare capacity preconfiguration for faster restoration of mesh networks", Submitted to Journal of Networks and Systems Management, Special Issue on Fault Management in Communication Networks, March 1996.

[51] McDysan, D. E., Spohn, D. L., ATM Theory And Application, New York, NY: McGraw-Hill Inc., 1995.

[52] MIL 3, Inc., *OPNET Modeler Manuals*, 3400 International Drive NW, Washington DC, 1994.

[53] Murakami, K., Kim, H., "Joint optimization of capacity and flow assignment for self-healing ATM networks", *Proc. IEEE ICC'95*, June 1995, pp. 216-220.

[54] Pekarske, B., "1.5 second rstoration using DS-3 cross-connects", *Trends in Network Restoration Symposium*, Edmonton, Canada, May 24-25, 1990.

[55] *Restoration of DCS mesh networks with distributed control: Equipment Framework Generic Criteria*, FA-NWT-001353, Issue 1, Bellcore, Dec. 1992.

[56] Rosen, E.C., "The updating protocol of ARPANET's new routing algorithm", Computer Networks, vol. no. 1, February 1980, pp. 11 - 30.

[57] Sakauchi, H., Nishimura, Y., Hasegawa, S., "A self-healing network with an economical spare-channel assignment", *Proc. IEEE Globecom '90*, Dec. 1990, pp. 438-443

[58] Saniee, I., "Optimal routing designs in self-healing communications networks", *Bellcore, MRE 2D-362*, 445 South Street, Morristown, NJ 07960-6438, fourth draft, May 1994.

[59] Sato, K., Okamoto, S., Hadama, H., "Network performance and integrity enhancement with optical path layer technologies", *IEEE J-SAC Special Issue: Integrity of Public Telecommunication Networks*, vol. 12, no. 1, Jan. '94, pp. 159 - 169.

[60] Schwartz, M., Stern, T. E., "Routing techniques used in computer communications", *IEEE Transactions on Communications*, vol. 28, no. 4, April 1980, pp. 539 - 552

[61] Sosnosky, J., "Service applications for SONET DCS distributed restoration", *IEEE J-SAC Special Issue: Integrity of Public Telecommunication Networks*, vol. 12, no. 1, Jan. '94, pp. 59-68.

[62] Struyve, Kris, Demeester, P., "Design of distributed restoration algorithms for ATM meshed networks", *Proc. of the 1995 IEEE 3rd Symposium on Communications and Vehicular Technology*, 1995, pp. 128 - 135.

[63] *The Role of Digital Crossconnect Systems in Transport Network Survivability*, SR-NWT-002514, Issue 1, Bellcore Special Report, Jan. 1993

[64] Topkis, D.M., "A k shortest path algorithm for adaptive routing in communications networks", *IEEE Transactions on Communications*, vol. 36, no. 7, July 1988, pp. 855 - 859.

[65] Veerasamy, J., Venkatesan, S., "Effect of traffic splitting on link and path restoration planning", *Proc. IEEE GlobeCom '94*, Dec. 1994, pp. 1867 - 1870.

[66] Venables, B.D., *Algorithms for the Spare Capacity Design of Mesh Restorable Networks*, Master of Science Thesis, University of Alberta, Fall, 1992

[67] Venables B. D., Grover, W. D., MacGregor, M. H., "Two strategies for spare capacity placement in mesh restorable networks", *Proc. IEEE ICC'93*, May 1993, pp. 267-271.

[68] *Wideband and Broadband Digital Cross-Connect Systems Generic Criteria*, FR-NWT-000440, Issue 3, Bellcore, Nov. 1993.

[69] Wu, T., *Fiber Network Service Survivability*, Norwood, MA: Artech House Inc., 1992.

[70] Wu, T., Kobrinski, H., Ghosal, D., Lakshman, T. V., "The impact of SONET digital cross-connect system architecture on distributed restoration", *IEEE J-SAC Special Issue: Integrity of Public Telecommunication Networks*, vol. 12, no. 1, Jan. '94, pp. 79 - 87.

[71] Yang, C.H., Hasegawa, S., "FITNESS: Failure immunization technology for network service survivability", *Proc. IEEE Globecom '88*, Dec. 1988, pp. 47.3.1-47.3.5